Title

Enhancing Estimation and Uncertainty Quantification in Stochastic Optimization: Importance Sampling and Bootstrap Resampling

Author

Chen, Xiaotie

Publication Date

2024

Peer reviewed|Thesis/dissertation

Enhancing Estimation and Uncertainty Quantification in Stochastic Optimization: Importance Sampling and Bootstrap Resampling

By

Xiaotie Chen

DISSERTATION

Submitted in partial satisfaction of the requirements for the degree of

DOCTOR OF PHILOSOPHY

in

APPLIED MATHEMATICS

in the

OFFICE OF GRADUATE STUDIES

of the

UNIVERSITY OF CALIFORNIA

DAVIS

Approved:

David L Woodruff, Chair

Jesus De Loera

Thomas Strohmer

Committee in Charge

2024

Contents

Abstract

This dissertation explores the applications of Monte Carlo and Bootstrap methods in stochastic optimization, focusing on enhancing computational efficiency and accuracy in solution evaluation and uncertainty quantification. For the purpose of computing the expected value of a stochastic optimization problem via simulation, we propose a method to efficiently construct importance sampling distributions using surrogate modeling. This method significantly reduces the need for repeated evaluations of the objective function, which are typically computationally intensive due to the reliance on optimization algorithms. Our method outperforms traditional Monte Carlo estimation and achieves significant speed-ups with good parallel efficiency. Additionally, we explore Monte Carlo sampling algorithms that utilize known distributions to construct confidence intervals around the optimality gap in two-stage and multi-stage stochastic programs. In scenarios where the distribution of uncertainty remains unknown, we discuss bootstrap and bagging algorithms that rely solely on sampled data to provide both a consistent sample-average solution and accurate confidence interval estimates. These methods are enhanced by integrating distribution estimation into resampling techniques to improve the precision of estimations under uncertainty. We also offer open-source software implementations of these algorithms. Extensive empirical studies show the effectiveness of the smoothed bootstrap and bagging methods, particularly for constructing confidence intervals with small data sets.

Acknowledgments

First and foremost, I would like to express my deepest gratitude to my advisor, Prof. David L. Woodruff, for his guidance, support, and understanding over the past several years. His patience and insight have profoundly shaped my entire graduate study experience. Without his invaluable mentorship, the completion of this dissertation would not have been possible.

I am also grateful to Prof. Jesus De Loera and Prof. Thomas Strohmer for their invaluable service on my dissertation committee, and for the help and suggestions I have received from them throughout the years. Additionally, I would like to express my gratitude to Prof. Miles Lopes for introducing me to the Bootstrap world and for his early guidance in my Ph.D. career.

I would like to extend my gratitude to the colleagues and friends I met during my studies at Davis for their support at various stages of my PhD journey. I am particularly thankful for the friendships I have developed with Joseph Pappe and Dongmin Roh. I am truly lucky to have met you both. My PhD experience would have been completely different without your constant companionship and support.

I am also deeply grateful to my dearest friends: Yiqi Wang, Yiwen Wang, Wenqi Wu, and Xinyuan Zhang. Despite the long distance between us, I cherish every moment we have spent together and all the conversations we have shared over the years.

And although he may not understand the extent of his contribution, I must extend my heartfelt thanks to my cat BoBo. His mere presence, comforting purrs, and patient companionship have been a pillar of emotional support during some of the most challenging moments of this journey, especially throughout the pandemic.

Lastly, special thanks go to my partner, Weidi Zhang. Thank you for your love and for being by my side. And to my family, my constant source of inspiration and support. Your endless encouragement and unconditional love have shaped who I am - I am here because of you.

CHAPTER 1

Introduction

1.1. Motivation and Background

Stochastic programming (SP) involves modeling optimization problems under uncertainty. The uncertainty may arise from the variability in data inputs or the absence of essential data, such as the future product demands, when decisions are required. The field of stochastic optimization has been widely applied across various sectors, including energy planning [79, 100], supply chain management [3, 88], finance [103], and transportation and logistics [36, 81], and there has been a relatively rich literature on how to formulate and solve a stochastic program [10, 52, 82, 87].

An illustrative example is the two-stage stochastic programming model, which can be formulated

$$\begin{aligned} z^* = \min_{x} \quad & E_\xi[g(x,\xi)], \\ s.t. \quad & x \in X, \end{aligned} \tag{1.1}$$

where $g(x,\xi)$ is the optimal value of the second-stage problem

$$\begin{aligned} g(x,\xi) = \min_{y} \quad & h(x,\xi), \\ s.t. \quad & y \in Y(x,\xi), \end{aligned} \tag{1.2}$$

here X and $Y(x,\xi)$ denote the feasible regions for the first stage solution x and the second stage solution y, respectively.

The two-stage stochastic programming makes decisions in two phases: the first stage occurs before the uncertainty is revealed, where a decision is made based on available information. The second stage takes place after the uncertainty has been revealed, allowing for adjustments to the initial decision based on the outcomes of the uncertain parameters.

In practical applications, it is often unrealistic to enumerate all possible scenarios due to the vast scale of the problem or the inaccessibility of the underlying parameter distributions. Consequently, it becomes necessary to approximate the stochastic optimization problem. To this end, recent

research has increasingly explored sampling methods to computationally address problems that are technically intractable.

For example, in situations where the problem parameters are random variables with known distributions, the sample average approximation (SAA) approach [54, 80, 84] has been employed to approximate the problem. The SAA method involves generating a finite number of scenarios according to the data distribution, then solving the deterministic optimization problem for these scenarios. The SAA formulation for the two-stage stochastic programming problem is given by

$$\begin{aligned} \min_{x} \quad & \frac{1}{N}\sum_{i} g(x,\xi_i), \\ s.t. \quad & x \in X, \end{aligned}$$

where ξ_i are samples drawn from the distribution of ξ, and the function g is as defined in (1.2). When ξ_i are independently and identically distributed (i.i.d.), this approximation is referred to as the standard Monte Carlo approximation.

The discussion of asymptotic properties of the SAA solutions obtained can be found in works such as [28, 31, 92, 94]. In addition, a series of papers [20, 21, 49] have discussed the importance sampling scheme for estimating the function value. In terms of uncertainty quantification, studies [24, 64, 67] use sampling methods to construct confidence intervals for the optimality gap of a given candidate solution; meanwhile, [7, 8, 41] have developed sequential sampling methods that produce a series of candidate solutions and estimate the quality of the corresponding solution.

For the integration of sampling within optimization algorithms, [46] and [74] incorporate sampling within the branch-and-bound algorithm and use stochastic upper and lower bound estimators to prune the search tree. A stochastic cutting plane method detailed in [44] employs sampling-based cutting planes within the L-shape method for stochastic programming. Additionally, [43] discusses sampling methods for assessing solution quality and developing stopping rules in the context of stochastic cutting plane.

While sampling methods are effective for situations with known probability distributions, challenges arise when the underlying distributions of random variables are unknown. In these cases, statistical inference must depend entirely on available data samples. This is where bootstrap methods come into play.

Bootstrap techniques utilize resampling from a single dataset to generate multiple simulated samples. It offers a practical alternative to traditional Monte Carlo-based statistical inference methods, especially when dealing with i.i.d. samples from an unknown distribution. The bootstrap method operates by initially drawing a random sample from the original dataset, which acts as a miniature representation of the entire population. Each bootstrap sample is then generated by randomly selecting observations from this initial sample with replacement. This process is repeated to construct a distribution of bootstrap samples. If the initial sample is adequately representative, then multiple resamplings effectively simulate the process of drawing multiple samples directly from the original population. Consequently, the statistical estimates obtained from these bootstrap samples can approximate those of the true distribution.

Since introduced in [29], bootstrap methods have been widely used for statistical inference; see [22, 30, 91] for a comprehensive introduction. In the area of stochastic programming, an early work of [41] used bootstrap to develop stopping rules for the Stochastic Decomposition algorithm. Eichhorn and Romisch [31] and Lam and Qian [59] proposed the use of bootstrap and related resampling methods to derive confidence intervals for the optimal function value. Anitescu and Petra [2] discussed some of the theoretical properties of the bootstrap confidence interval for stochastic programming.

1.2. Contributions and Organization

The dissertation is structured as follows. Chapter 2 is based on the work in [17]. In Chapter 2, we explore the Monte Carlo (MC) method and variance reduction techniques that have been used to estimate the function value for a candidate solution when the underlying distribution of the random variable is known. We propose a method for efficiently constructing importance sampling distributions using surrogate modeling. This method addresses the computational challenges encountered with the traditional MC method by minimizing the number of evaluations required for the objective function and leveraging parallelism to significantly reduce runtime. We also include experimental results to support the proposed method. In Chapter 3, we introduce Monte Carlo sampling algorithms that construct confidence intervals around the optimality gap for two-stage and multi-stage stochastic programming when an unlimited number of scenarios can be sampled. This chapter is based on the pre-print [14]. In Chapter 4, we describe bootstrap and bagging methods

for stochastic programming that use only sampled data to obtain both a consistent sample-average solution and a consistent estimate of confidence intervals for the optimality gap. The underlying distribution from which the samples are drawn is not required. In addition, we introduce innovations that integrate distribution estimations with resampling techniques to improve estimations for optimization problems under uncertainty. In Chapter 5, we describe the software we developed for data-based stochastic programming using bootstrap and bagging methods as outlined in Chapter 4. The software is open source and available on Github. We also provide extensive numerical results for method comparison. The methods, software, and comprehensive numerical results showcased here are informed by our work in [**15**, **16**]. Finally, in Chapter 6, we conclude the dissertation and address the remaining questions.

CHAPTER 2

Recourse Function Estimation: Importance Sampling using Surrogate Modeling

2.1. Introduction

The evaluation of the function value given a first-stage candidate solution $\widehat{x}$ is crucial in stochastic programming. By carefully assessing this function value, decision makers can effectively quantify the expected costs, benefits, or risks associated with the initial decision $\widehat{x}$, taking into account various scenarios of uncertainty. Furthermore, the iterative steps in the decomposition methods for optimizing stochastic programming, such as Stochastic Dual Dynamic Programming (SDDP) [79] and the L-shape method [99], rely heavily on the effective evaluation of the recourse function. Specifically, in SDDP, this evaluation is central to updating the value function approximations, while in the L-shaped method, it is essential for generating and refining Benders cuts that iteratively improve the solution quality of the master problem.

In this chapter, we address the problem of evaluating the function value of a stochastic programming problem stated abstractly as

$$z^* = \min_x E_{p(\xi)}[g(x, \xi)],$$

where constraints are implicitly incorporated into the function values, and the function g is designed to account for complex structures and future states. The distribution of the random variable ξ is denoted by $p(\xi)$. Given a candidate solution $\widehat{x}$, our objective is to determine the expected function value:

$$\widehat{z} = E_{p(\xi)} g(\widehat{x}, \xi). \tag{2.1}$$

2.2. Monte Carlo Method for Stochastic Optimization

The evaluation of the expectation in (2.1) can be computationally challenging and time-consuming, as it often involves nested optimization problems within an integral. Moreover, obtaining exact solutions for these integrals is frequently intractable in practical settings, especially when the random variables are continuous. In such situations, Monte Carlo estimation is often employed as an effective alternative. Monte Carlo estimation offers a practical method for approximating the function when exact solutions are elusive. It utilizes random sampling to provide estimates that converge to the true value as the number of samples increases. A standard Monte Carlo method involves taking a set of N samples of the random variable $\xi_1, \ldots, \xi_N$ and approximating the integral $\widehat{z}$ with:

$$\widehat{z}^{MC} = \frac{1}{N} \sum g(\widehat{x}, \xi_i) \tag{2.2}$$

It is well-known that the standard error of such Monte Carlo estimators can be expressed as

$$SE(\widehat{z}^{MC}) = \frac{\sigma_{g(\widehat{x},\xi)}}{\sqrt{N}},$$

where $\sigma_{g(\widehat{x},\xi)}$ represents the standard deviation of the function g at $\widehat{x}$. Therefore, when the function g exhibits high variance, the basic Monte Carlo method can require a large number of samples to achieve the desired level of precision, with each sample corresponding to a computationally expensive optimization problem.

Many variance reduction techniques have been applied alongside the basic Monte Carlo method to enhance its efficiency and accuracy [46], including:

- Control Variates: This method utilizes an additional variable, known as the control variate, with a known expected value. By establishing a correlation between this control variate and the function g, it is possible to refine the estimate by adjusting for the difference between the known expected value of the control variate and its observed average. With a carefully chosen control variate and an appropriate scaling factor γ, the variance in the estimate can be reduced [40, 71].
- Stratified Sampling: Stratified sampling first divides the domain into different strata, then draws samples from each strata to ensure a more uniform and representative distribution of samples across the entire domain. A classic approach within stratified sampling is the

Latin Hypercube Sampling, where the domain of each input variable is divided into equally probable intervals. Samples are then selected so that each interval of every variable has exactly one sample [33, 45, 68].

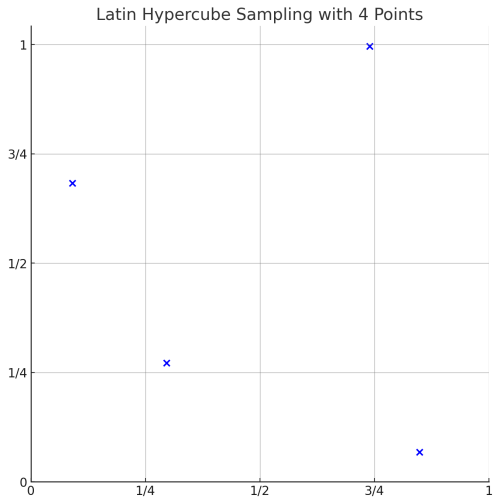

FIGURE 2.1. Latin Hypercube Sampling in two dimensions with 4 Points

- Quasi-Monte Carlo: Quasi-Monte Carlo methods employ low-discrepancy sequences, such as Halton and Sobol sequences, to reduce the gaps and clusters typically found in random samples. This approach results in a more evenly distributed set of samples across the domain, hence reducing the variance in the estimation of integrals [27, 57, 62, 73].
- Importance Sampling: Importance sampling involves a change of variables and the adoption of a different probability distribution, referred to as the importance sampling distribution and denoted as $q(\xi)$. In this way, the function value $\hat{z}$ in (2.1) can be expressed as:

$$\begin{aligned} \hat{z} &= E_{p(\xi)} g(\hat{x}, \xi) \\ &= \int g(\hat{x}, \xi) p(\xi) d\xi \\ &= \int \frac{g(\hat{x}, \xi) p(\xi)}{q(\xi)} \cdot q(\xi) d\xi \\ &= E_{q(\xi)} \frac{g(\hat{x}, \xi) p(\xi)}{q(\xi)}. \end{aligned}$$

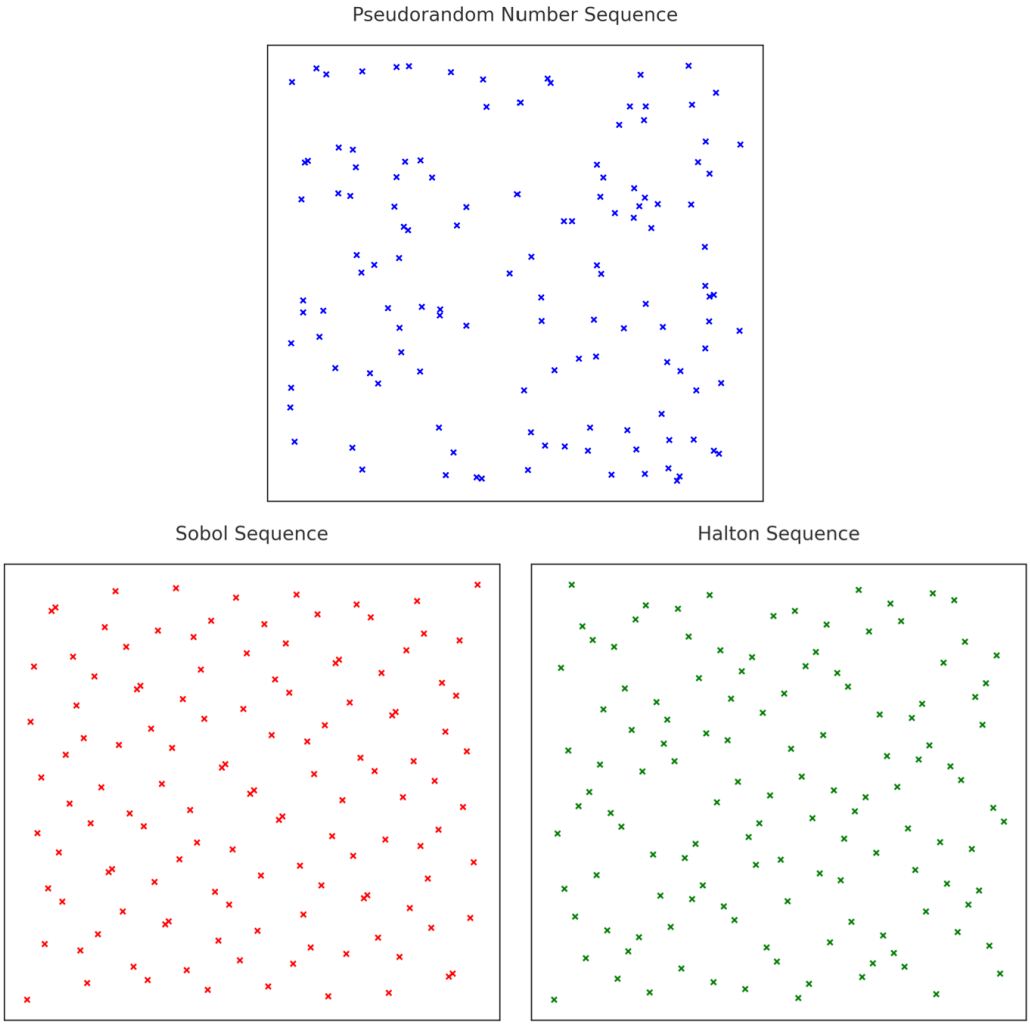

FIGURE 2.2. Comparison of random sequences: Pseudorandom, Sobol, and Halton sequences, each generated with 256 points. The Sobol and Halton sequences exhibit reduced clustering and a more uniform distribution compared to pseudorandom sequences.

Then, instead of using (2.2) for evaluation, we can generate samples from this importance distribution $q(\xi)$ and construct our estimate as follows:

$$\hat{z}^{IMC} = \frac{1}{N} \sum_i \frac{g(\hat{x}, \xi_i) p(\xi_i)}{q(\xi_i)} \tag{2.3}$$

In theory, as long as the support of the original probability p is adequately covered by the support of the importance sampling distribution q (i.e., $p(\xi) > 0 \implies q(\xi) > 0$),

utilizing Equation (2.3) will yield an unbiased estimate. Nonetheless, selecting a suitable q is crucial to avoid the need for excessively large estimation sample sizes.

The Monte Carlo method has long been used in stochastic programming to approximate the function values [10, 46], with most of them focusing mainly on the basic Monte Carlo method [54, 92]. The literature on importance sampling in the area of stochastic programming starts with a series of papers [20, 21, 49] that introduced an importance sampling scheme, based on strong assumptions. They assumed that the cost surface can be approximated using an additive, separable model that considers marginal costs in each dimension alongside a base cost. Under these assumptions, they constructed importance sampling distributions for each dimension and aggregated the estimates from these dimensions to derive the final approximation.

Parpas et al. [76] proposed a method that does not rely on the assumptions of additive models. They directly generated samples from the optimal distribution $q^*(\xi)$ in Equation (2.4) using a Markov Chain Monte Carlo (MCMC) algorithm. Subsequently, they used these samples to "recover" or reconstruct an approximation of $q^*(\xi)$ using a kernel density estimation (KDE) algorithm, resulting in $\widehat{q}_M$. Once this approximation was reconstructed, Equation (2.3) could be employed to approximate $\widehat{z}$. However, the limitation of the MCMC-IS method [76] is that it often requires a large number of samples generated by the MCMC algorithm to obtain a good approximation of q^*. Furthermore, many of the generated samples are discarded, which can be computationally wasteful.

In Section 2.3 and 2.4, we propose a method for efficiently constructing importance sampling distributions using surrogate modeling. We then utilize these specially designed distributions to estimate the function value $\widehat{z}$. A software implementation of the methods called SMAIS is available at https://github.com/DLWoodruff/SMAIS.git. Our approach aims to address these key considerations:

- Efficient use of Function Evaluations: Our methods are designed to minimize the wasted evaluation of the function $g(\widehat{x}, \xi)$. Since these evaluations often require the use of optimization algorithms and can be computationally expensive, minimizing their computation is crucial for efficient estimation.
- Parallelism: We aim to exploit parallelism to achieve reductions in run-time.

Our ideas are inspired by techniques in surrogate optimization; see [37] for an overview. Surrogate optimization deals with scenarios where function evaluations can be even more computationally

demanding compared to our context. Additionally, our work has connections to response surface methodology, where function evaluations sometimes involve physical experiments, but not always.

In [4] a response surface methodology (RSM) was proposed for the sensitivity analysis of two-stage stochastic programming problems. In particular, the RSM enables efficient identification of the sensitivity to changes in first-stage variables. The paper uses a quadratic response surface and reports good results. Latin hypercube sampling is advocated as a variance reduction technique.

The idea of combining surrogate model with importance sampling was previously explored in the area of engineering design optimization, mostly focused on analyzing the failure rate. Peherstorfer et al. [104] uses importance sampling with surrogate models, with a specific application to failure rate/yield estimation in certain systems. See also [77, 97]. However, the methods developed for failure rate estimation do not directly translate to stochastic programming, as they primarily focus on generating a distribution for the failure region, while in stochastic programming the entire region needs to be considered for optimization.

2.3. Importance Sampling with Surrogate Modeling

The selection of an appropriate $q(\xi)$ can significantly reduce the variance of the estimator. It has been demonstrated that the optimal importance sampling distribution takes the form:

$$q^*(\xi) = \frac{|g(\widehat{x},\xi)|p(\xi)}{E_{p(\xi)}|g(\widehat{x},\xi)|}. \tag{2.4}$$

In scenarios where the function g is non-negative, it simplifies to

$$q^*(\xi) = \frac{g(\widehat{x},\xi)p(\xi)}{\widehat{z}},$$

making $q^*(\xi)$ a zero-variance estimator. This implies that a perfect estimate of $\widehat{z}$ could theoretically be obtained with just a single observation ξ_0, as

$$\widehat{z} = \frac{g(\widehat{x},\xi_0)p(\xi_0)}{q^*(\xi_0)}.$$

Nevertheless, it is essential to note that determining the exact form of $q^*(\xi)$ directly is impractical without knowing beforehand the exact value of $\widehat{z} = E_{p(\xi)}|g(\widehat{x},\xi)|$, which is precisely what we aim to evaluate in the first place.

As a workaround, we seek to construct an importance sampling distribution q that closely mirrors the ideal q^*, particularly targeting areas where $|g(\widehat{x}, \xi)|$ exhibits significant values. A straightforward strategy to achieve this involves iteratively computing the value $|g(\widehat{x}, \xi)|p(\xi)$ on various realizations of ξ; then we use these computed values to construct an approximate distribution. However, this direct computation can be quite resource intensive, especially when dealing with complex functions such as $g(\widehat{x}, \xi)$.

To address this computational challenge, we leverage Surrogate Modeling techniques. Surrogate models are simplified and computationally efficient approximations that mimic the behavior of the original complex and often expensive-to-evaluate models (the function g in our context). They are commonly used in experiments and simulations in engineering design to speed up the decision-making process and reduce the computational burden. Examples of common approximation techniques include the following:

- Polynomial Response Surfaces (PRSM): PRSM models the response of a function using a polynomial equation of its input variables. It constructs a surface that approximates the target function across the input space by fitting polynomial terms of varying degrees, where the choice of degree depends on the complexity and the nature of the relationship between inputs and outputs.
- Radial Basis Functions (RBF): The RBF method models the target function as a linear combination of basis functions, each associated with a point in the training set. The value of the basis function depends solely on the distance between the prediction point and the corresponding training point.
- Kriging (KRG): In Kriging, the model is formed as a weighted sum of observed values. The weights are determined by the proximity and spatial configuration of the data points in relation to the prediction location. These weights are calculated from a spatial correlation model derived from the observed data.
- Artificial Neural Networks (ANN): ANNs model the relationship between inputs and outputs through layers of interconnected nodes or neurons, where each connection represents a weight. The network learns by adjusting these weights to minimize the error between the predicted and actual outputs.

For a comprehensive exploration of Surrogate Modeling techniques, including those not listed here, readers are encouraged to refer to [37].

To effectively construct an importance sampling distribution that approximates the optimal sampling function q, we create a surrogate model, referred to as s. This surrogate model s is used in place of the function g to construct an importance sampling distribution q_s, such that

$$q_s(\xi) = \frac{|s(\xi)|p(\xi)}{\int_\Omega |s(\xi)|p(\xi)}. \tag{2.5}$$

In particular, q_s exhibits structural similarities to q^* in Equation (2.4). In essence, $s(\xi)$ acts as a substitute for $g(\xi)$ in the importance sampling distribution.

The importance sampling with surrogate modeling (*SM-IS*) procedure for estimating $\widehat{z}$ can be outlined as follows:

(1) **Training Sample Generation**: Generate a set of training samples, $\{\xi_i\}_{i=1}^M$, using stratified sampling methods such as Latin Hypercube Sampling. A diverse set of samples is important for achieving accurate surrogate modeling.

(2) **Surrogate Model Construction**: For each training sample, compute $y_i = g(\widehat{x}, \xi_i)$. With the input-output pairs (ξ_i, y_i), construct the surrogate model s.

(3) **Importance Sampling Distribution Creation**: Use the surrogate model s to craft an importance sampling distribution, denoted as q_s, defined as follows:

$$q_s(\xi) = \frac{|s(\xi)|p(\xi)}{\int_\Omega |s(\xi)|p(\xi)}, \tag{2.6}$$

where the denominator's integration is estimated using Monte Carlo methods. This estimation relies solely on s without the need for direct access to g. Given that evaluating $s(\xi)$ is generally much more computationally economical than evaluating $g(\xi)$, the computation of the denominator is correspondingly cost-effective.

(4) **Final Estimation**: Use rejection sampling, as discussed in Section 2.3.1, to draw N points $\{\xi_1, \ldots, \xi_N\}$ from the distribution q_s. Estimate $\widehat{z}$ with the sampled points using

$$\widehat{z}^{IMC} = \frac{1}{N} \sum_i \frac{g(\widehat{x}, \xi_i)p(\xi_i)}{q_s(\xi_i)}$$

This step involves evaluating the original function g. A more accurate approximation of q_s to q^* can reduce the number of samples needed to obtain a stable estimate.

2.3.1. Sampling From the Constructed Distribution. To estimate the function value using the importance sampling distribution constructed in Step 3 of the SM-IS procedure,

$$q_s(\xi) = \frac{|s(\xi)|p(\xi)}{\int_\Omega |s(\xi)|p(\xi)},$$

it is necessary to draw samples from $q_s(\xi)$. Given that the constructed sampling distribution involves the surrogate model s and may take a complicated form, direct sampling may not be feasible. Therefore, we employ rejection sampling. The rejection sampling method facilitates drawing random samples from a target distribution that is difficult to sample from directly, especially when the distribution lacks a closed form or is difficult to work with.

The core idea of rejection sampling is to initially sample from a simpler, auxiliary distribution π, which can be easily simulated. Then each sampled point is accepted or rejected based on a probability proportional to the ratio of the target probability density to the proposal probability density, multiplied by a normalization constant κ. This constant κ is chosen as the smallest number such that $\kappa\pi(\xi) \geqslant q_s(\xi)$ for all ξ, thus ensuring that the scaled proposal distribution covers the target distribution. In this way, the accepted samples conform to the target distribution.

For simplicity, we adopt a uniform distribution as our auxiliary distribution π, from which the proposal samples are drawn randomly. Instead of traditional quasi-random proposal samples for acceptance-rejection, [34] propose the use of Generalized Fibonacci Lattices. These lattices ensure that the proposal samples cover the state space more uniformly and help to avoid the frequent clusters and gaps that occur with random samples. In practice, other low-discrepancy sequences, such as Sobol and Halton sequences, have also demonstrated better coverage compared to random sequences, as shown in Section 2.2.

The subroutine for drawing samples from the constructed importance sampling distribution $q_s(\xi)$ using rejection sampling with low-discrepancy sequences is outlined for completeness. We assume that the domain of the random variable ξ is a multi-dimensional box $\mathcal{B}$. For detailed methodology, we refer readers to [34].

(1) **Generate Samples:** Draw a large number K of samples from the multi-dimensional box $\mathcal{B}$ using low-discrepancy sequences.

(2) **Evaluate Integral:** Evaluate the integral $\int_\Omega |s(\xi)|p(\xi)\, d\xi$ for $q_s(\xi)$ and compute the maximum value, denoted M, of $q_s(\xi)$ using the K samples.

(3) **Compute Scaling Factor:** Compute the rejection scaling factor κ as

$$\kappa = M \times \text{Area}(\mathcal{B}).$$

(4) **Proposal Sampling:** Independently draw a sample $\widehat{\xi}$ using a low-discrepancy sequence from $\mathcal{B}$. In addition, generate a uniform random number u from 0 to 1.

(5) **Accept or Reject:** If $u \leqslant \frac{q_s(\widehat{\xi})}{\kappa\pi(\widehat{\xi})}$, accept $\widehat{\xi}$ as a sample from q_s; otherwise, reject it and repeat the process.

2.4. Adaptive Importance Sampling with Surrogate Modeling

In the SM-IS algorithm, both the size of the training sample set in Step 1 and the size of the evaluation sample set in Step 4 are explicitly predetermined. The number of evaluation samples used in Step 4 directly affects the quality of the final estimator $\widehat{z}^{IMC}$. However, finding a suitable sample size that balances accuracy and computational efficiency depends on the quality of the surrogate model s, which is determined by both the quantity and quality of the training samples drawn in Step 1. Inadequate sampling in Step 1 may yield a suboptimal approximation of the function g, while an excessive number of samples can lead to an inefficient use of computational resources.

Instead of adhering to a fixed number of training and evaluation samples, an adaptive, multi-fidelity approach can be more beneficial. This means that during the surrogate model construction step, we increase the number of samples based on the quality of the current constructed surrogate model, until we build an acceptable one. Similarly, in the final estimation step, we may adaptively draw samples from the constructed importance sampling distribution q_s, until the estimation $\widehat{z}^{IMC}$ converges. The idea of adaptive sampling for surrogate modeling has continuously drawn attention in the engineering community [66, 72, 101, 102], where most of them are designed for specific applications and do not apply directly in the stochastic programming setting. Adaptive sampling unrelated to importance sampling has been explored in stochastic programming, by, e.g., [42] and [9].

We provide a general framework for adaptive importance sampling with surrogate modeling (SM-AIS) in the following algorithm, and discuss the criteria for assessing the efficacy of the constructed surrogate model and for determining the appropriate stopping rule in later subsections.

1. **Initialization**:
   - Initialize the iteration index, $k = 0$
   - Generate the initial training sample set $\Xi^0 = \{\xi_i\}_{i=1}^M$ (e.g., using Latin hypercube sampling).
   - Evaluate $y_i = g(\widehat{x}, \xi_i)$ for each training sample.
2. **Construct Surrogate Model**:
   - Construct a surrogate model s^k mapping ξ_i to $y_i = g(\widehat{x}, \xi_i)$ for each sample in the training set Ξ^k. The $\{y_i\}$ values have already been obtained in the previous step(s).
3. **Construct Importance Sampling Distribution** q_s^k:
   - Construct the current importance sampling distribution as:

$$q_s^k(\xi) = \frac{|s^k(\xi)|p(\xi)}{\frac{1}{K}\sum_j |s^k(\widetilde{\xi}_j)|p(\widetilde{\xi}_j)}, \tag{2.7}$$

   Here $\{\widetilde{\xi}_j\}$ are some random samples drawn from the domain of the random variable ξ using the regular Monte Carlo method or the quasi-Monte Carlo method. The denominator in the expression of $q_s^k(\xi)$ serves as an approximation of the integral $\int_\Omega |s^k(\xi)|p(\xi)$ that determines the proper scaling of the importance sampling distribution. Although an accurately estimated integral of $\int_\Omega |s^k(\xi)|p(\xi)$ is essential, this estimation, involving only the surrogate model s^k, is generally not computationally intensive compared to calculations involving g.
4. **Assess Surrogate Model Quality**:
   - Assess the quality of the surrogate models as discussed in Section 2.4.1.
   - If the quality of the surrogate model satisfies the predetermined stopping criteria, proceed to Step 6.: Final Estimation.
   - Otherwise, continue to the next step: Refine the Surrogate Model.
5. **Refine Surrogate Model**:
   - Choose some additional training samples. Add these to the training set Ξ^k to form a new training set Ξ^{k+1}. The choice of the additional training samples is discussed in Section 2.4.2.
   - Increase the iteration index: $k = k + 1$
   - Return to step 2.

6. **Final Estimation**:
   - Adaptively sample from q_s^k using rejection sampling as detailed in Section 2.3.1 to collect evaluation samples $\{\xi_i\}$ and periodically estimate the function value $\widehat{z}$ with Equation (2.3). Continue until convergence is achieved. The stopping criteria are detailed in Section 2.4.3.

This method follows the principles of multi-fidelity approaches. Initially, we generate a low-fidelity estimate of the importance sampling distribution q_s to accelerate the computational process. Subsequently, for our final estimate (2.3), we use a high-fidelity assessment to ensure an unbiased approximation. The concept of multi-fidelity models has seen application in different areas [70, 78]. More recently, [1] discusses the theoretical aspect of multi-fidelity importance sampling.

2.4.1. Surrogate Model Assessment. Evaluating the quality of surrogate models is crucial in their development and refinement, as it directly impacts the accuracy and reliability of the estimates. We propose two approaches for assessing the quality of the surrogate models. The first approach relies on quantifying the error of the surrogate model, offering a direct measure of the model's accuracy in replicating the true responses. The second approach is based on the estimation of the variance of the final estimator, which indicates the consistency and stability of the estimator.

2.4.1.1. *Direct Error-Based Assessment.* To construct an effective surrogate model, the primary goal is to ensure that its associated importance sampling distribution,

$$q_s^k(\xi) = \frac{|s^k(\xi)|p(\xi)}{\frac{1}{K}\sum_j |s^k(\widetilde{\xi}_j)|p(\widetilde{\xi}_j)},$$

aligns with the optimal sampling function

$$q^*(\xi) = \frac{|g(\xi)|p(\xi)}{\int_\Omega |g(\xi)|p(\xi)}.$$

Given this objective, a direct and natural approach for surrogate model assessment involves evaluating the maximum absolute error, or the $L^{\inf}$ norm, between the surrogate function s and the true function g. More specifically, we focus on quantifying the discrepancy between the function $|s^k(\xi)|p(\xi)$ and $|g(\xi)|p(\xi)$,

$$\epsilon_s = \sup_{\xi\in\Omega} \big||s^k(\xi)|p(\xi) - |g(\xi)|p(\xi)\big|,$$

as a measure of the accuracy of the surrogate model. Our selection of absolute error over relative error is in accordance with the principles of importance sampling. This choice allows a greater tolerance for errors in areas of lesser significance, i.e. areas with lower q values, thereby focusing our computational resources and efforts on areas with high q values, ensuring that our surrogate model achieves a high degree of precision in important regions.

To estimate the error ϵ_s, we draw a manageable set of samples and compute the corresponding absolute error between $|s(\xi_j)|p(\xi_j)$ and $|g(\xi_j)|p(\xi_j)$ for each sampled ξ_j. The composition of these samples is twofold: A portion (of cardinality M_q) is drawn according to the importance sampling distribution of the surrogate model q_s, ensuring that the surrogate model's approximations are precise in areas it identifies as crucial. The remainder of the samples (of cardinality M_r) is drawn from a uniform distribution across the domain of the random variable ξ, which helps verify the model's overall performance and guards against over-fitting to the training samples.

The decision to refine the surrogate model s further is guided by a stopping criterion centered on whether the maximum absolute error across the assessment samples exceeds a predefined threshold. This threshold may be expressed either as a fixed absolute number or, as we have found effective in our experiments, in the form of

$$c_\beta \times \max |g(\xi_j)|p(\xi_j),$$

where c_β represents a predetermined scaling factor. Implementing a threshold in this manner simplifies the adjustment of the threshold value, aligning with our goal to develop an importance sampling distribution that emphasizes accuracy in critical regions characterized by high q_s values.

2.4.1.2. *Variance-Based Assessment.* An alternative method to evaluate the surrogate model focuses on examining the variance of the final estimator $\widehat{z}^{IMC}$. This strategy is based on the understanding that a high-quality surrogate model will yield a final estimator with lower variance. Consequently, assessing this variance serves as an indirect but effective way to gauge the accuracy of the surrogate model s and inform its further refinement. Specifically, with the surrogate model s and the associated importance sampling distribution q_s, the convergence rate of the final estimator $\widehat{z}^{IMC}$ in Equation (2.3) is expressed as $O\left(\frac{\sigma_s}{\sqrt{N}}\right)$, as $\widehat{z}^{IMC}$ is the average of N independent samples.

The variance σ_s^2 is determined by

$$\sigma_s^2 = E_{q_s}\left(\frac{g(\widehat{x},\xi)p(\xi)}{q_s(\xi)}\right)^2 - \left(E_{q_s}\frac{g(\widehat{x},\xi)p(\xi)}{q_s(\xi)}\right)^2. \tag{2.8}$$

Hence, we can develop a stopping criterion for the refinement of the surrogate model s, predicated on the variance σ_s^2 descending below a specified threshold.

To this end, we draw a modest collection of samples $\{\tilde{\xi}_i\}_{i=1}^{M_e}$ from q_s (of cardinality M_q). The variance σ_s^2 is then estimated either by direct computation:

$$\tilde{\sigma}_s^2 = \frac{1}{M_q - 1}\sum_i \left(\frac{g(\widehat{x},\tilde{\xi}_i)p(\tilde{\xi}_i)}{q(\tilde{\xi}_i)} - \tilde{z}\right)^2, \tag{2.9}$$

where

$$\tilde{z} = \frac{1}{M_q}\sum_i \frac{g(\widehat{x},\tilde{\xi}_i)p(\tilde{\xi}_i)}{q(\tilde{\xi}_i)}.$$

It should be emphasized that this approach may be more prone to over-fitting the training samples, making it appropriate only when a substantial initial training sample set is utilized, where there is sufficient coverage of regions considered potentially significant. In practical applications, combining variance-based evaluation with direct error-based assessment can yield the most effective results.

2.4.2. Additional Training Sample Selection. The selection of additional training samples is inherently informed by the assessment of the quality of the surrogate model. To maximize computational efficiency, our approach emphasizes the reuse of samples that have already contributed to the evaluation of the surrogate model. It is important to highlight that adding extra samples inevitably incurs some computational cost for retraining the surrogate model s. However, we avoid the need for repeated evaluations of the function value g, which can be costly, for these new samples, since these evaluations are already completed during the assessment phase.

As we want to allocate our computational resources to make significant accuracy improvements, we have explored two primary approaches for selecting additional training samples. The first involves selecting a fixed number or proportion of samples, identified during the assessment phase, that demonstrate the largest errors. The second method focuses on samples whose errors exceed a specific threshold in the form of $c_\beta \times \max|g(\xi_i)|p(\xi_i)$, as mentioned in Section 2.4.1.1. Through

practical application, we have observed that prioritizing samples with errors above a predetermined threshold, thus excluding those with minimal errors, results in a more balanced surrogate model.

2.4.3. Final Estimation and Confidence Interval Construction. In the final estimation step, we employ a dynamic process for sampling from the importance sampling distribution q_s, and periodically estimate the function value based on the collected samples. The estimator is given by:

$$\widehat{z}^{IMC} = \frac{1}{N}\sum_{i}\frac{g(\widehat{x},\xi_i)p(\xi_i)}{q_s(\xi_i)}.$$

Note that this approach allows for the simultaneous construction of confidence intervals for the true function value $\widehat{z}$ around the estimate $\widehat{z}^{IMC}$ using the Central Limit Theorem. These intervals are expressed as:

$$[\widehat{z}^{IMC} - t_{1-\alpha/2}\widetilde{\sigma}_s, \widehat{z}^{IMC} + t_{1-\alpha/2}\widetilde{\sigma}_s],$$

where $t_{1-\alpha/2}$ refers to the student-t distribution.

At the same time, it also facilitates the estimation of the total number of samples required to meet a specified error tolerance. Since the estimator $\widehat{z}^{IMC}$ is a sample average estimator, according to the central limit theorem, it converges to a normal distribution with mean $\widehat{z}$ and variance σ_s^2/N, at a rate of $O\left(\frac{\sigma_s}{\sqrt{N}}\right)$. Here, σ_s^2 is the associated variance as defined in (2.8).

In this way, To determine the sample size $N_{s,\epsilon}$ necessary for achieving an error tolerance ϵ, we use the formula

$$N_{s,\epsilon} = \left\lceil \left(\frac{Z_{1-\alpha/2}\cdot\sigma_s}{\epsilon}\right)^2 \right\rceil,$$

where $Z_{1-\alpha/2}$ is the z-score associated with a predefined confidence level. That is, with a sample size of $N_{s,\epsilon}$ and probability of $1-\alpha$, the error in the final estimator $\widehat{z}^{IMC}$ will be less than ϵ.

Although we do not have direct access to the variance σ_s, we can derive an initial estimation of the variance, $\widetilde{\sigma}_s$, from an early subset of samples drawn from the importance sampling distribution q_s. This preliminary variance estimation gives us a projection of the approximate number of samples, $\widetilde{N}_{s,\epsilon}$, to achieve our target precision.

As the adaptive sampling process progresses, each additional sample drawn incrementally refines our estimates of $\widetilde{\sigma}_s$ and $\widetilde{N}_{s,\epsilon}$. Simultaneously, it updates the confidence intervals for the true function value, $\widehat{z}$.

2.4.4. Parallelization of the Algorithm. The main algorithm lends itself well to parallelization, which can significantly reduce overall execution time. Here is an outline of the components of the algorithm that are suited for parallel execution.

- **Initialization:** The generation of the initial training sample set $\Xi^0 = \{\xi_i\}_{i=1}^M$ and the evaluation of their function values $g(\widehat{x}, \xi_i)$ can be easily parallelized, as the generation of each sample and the evaluation of the function value are independent of other samples. The initial sampling such as Latin hypercube sampling can be distributed across multiple processors, with each processor generating a subset of the total samples independently.
- **Construct Surrogate Model:** The potential for parallelizing this surrogate model training process largely depends on the type of surrogate model used and its parallelization capabilities. For instance, neural networks and certain implementations of Kriging can be trained in parallel by leveraging multiple processing units to handle batches of data or to compute model parameters concurrently. In cases where the surrogate model or the training methodology does not inherently support parallelization, alternative strategies, e.g. parallelizing the hyper-parameter tuning, can be explored.
- **Construct Importance Sampling Distribution q_s^k:** Constructing the importance sampling distribution q_s^k requires the evaluation of the value $|s^k(\tilde{\xi}_j)| * p(\tilde{\xi}_j)$ over a reasonably large set of samples, which can be processed in parallel as each calculation is independent. In our parallelization experiments, we allocated the computation of $|s^k(\tilde{\xi}_j)| \times p(\tilde{\xi}_j)$ among all accessible processors, subsequently gathering the outcomes to obtain the proper scaling of the importance sampling distribution q_s^k.
- **Assess Surrogate Model Quality:** The surrogate model's quality assessment involves the use of rejection sampling from q_s^k, a technique well-suited for parallel execution. In our experiments, each worker is tasked with examining B_a samples per batch, determining whether each sample should be accepted or rejected. This workflow is maintained across all workers until a collective total of M_q samples has been drawn from q_s^k.
- **Final Estimation:** The final estimation phase adopts a similar approach to the quality assessment step, employing parallelized rejection sampling from the importance sampling distribution, complemented by function evaluations. In our implementation, each worker is assigned the task of evaluating B_e samples within a batch to decide their acceptance.

For the samples that are accepted, the subsequent step involves calculating the weighted function value $\frac{g(\widehat{x},\xi_i)p(\xi_i)}{q_s(\xi_i)}$. The process is carried out uniformly by all workers, continuing until the stopping criteria are met. The final estimation of the function value $\widehat{z}$ is achieved by aggregating the calculated results from all contributing workers.

2.5. Experimental Results

In this section, we present the experimental results for the adaptive importance sampling with surrogate modeling (SM-AIS) method. To generate random candidate solutions $\widehat{x}$ for evaluating function values, we applied a sample average approximation to the original stochastic programming problems using a limited set of scenarios. Then each candidate solution $\widehat{x}$ was used to evaluate the efficacy of our SM-AIS method. Our proposed method was compared against two alternative approaches.

The first comparative method is the regular Monte Carlo (MC) method, wherein sampling is conducted according to the original probability distribution, denoted as p. We adopted rejection sampling with Sobol sequences to work with distribution p, allowing us to handle various types of distribution functions. Utilizing Sobol sequences enables more uniform sampling across the space and has shown greater efficiency than uniform random sampling, particularly in high-dimensional scenarios. To facilitate rejection sampling, we start by estimating a scaling factor. This involves preliminary sampling from p to approximate its maximum value, which we then utilize to derive an estimated scaling factor.

The second method is the Markov Chain Monte Carlo Importance Sampling (MCMC-IS) method, as proposed by [76]. This method involves generating samples from the optimal distribution $q^*(\xi)$ using MCMC and then approximating this distribution with kernel density estimation (KDE). We use a normal distribution as our proposal distribution, experimenting with both constant variance and adaptive Metropolis techniques [38]. Preliminary parameter tuning was conducted for the fixed-variance approach, and the best-performing setup was selected for method comparison.

For our proposed Adaptive Importance Sampling with Surrogate Modeling (SM-AIS) method, there is a wide option of surrogate models. In the experiments reported below, we adopt the kriging method because it leverages both the sampled data and the inherent correlation patterns among

the samples. Thus, it is capable of effectively approximating non-linear functions. Although we tested other types of surrogate models, we did not observe a significant difference in performance, provided that an ample number of samples is used for training.

2.5.1. Problem Examples. We conducted experiments over four different example problems.

2.5.1.1. *NewsVendor.* The NewsVendor problem involves finding the optimal order quantity to maximize profit or minimize costs while considering the balance between lost sales and excess inventory. In our experiment, we used the two-stage stochastic program example provided in [65], where the uncertain demand ξ follows a triangular distribution. In the first stage, the order quantity x is selected without knowing the actual demand ξ. Subsequently, in the second stage, decisions y_ξ are made regarding the quantity to sell to minimize the costs associated with inventory that remains unsold. The problem can be formulated as:

$$\begin{aligned} \underset{x, y_\xi}{\text{maximize}} \quad & x - 2x + \mathbb{E}_\xi[5y_\xi - 0.1(x - y_\xi)] \\ \text{subject to} \quad & y_\xi \leqslant x, \quad \forall \xi \in \Xi \\ & 0 \leqslant y_\xi \leqslant d_\xi, \quad \forall \xi \in \Xi \\ & x \geqslant 0. \end{aligned}$$

2.5.1.2. *CVaR.* CVaR, or Conditional Value at Risk, is a risk measure used in finance and statistics to quantify the potential loss in the worst-case scenarios of an investment or portfolio. It is defined with a parameter α, representing the expectation of losses in the worst α fraction of outcomes. In our experiment we solve a $(1-\alpha)$-level CVaR problem as in [59]:

$$\min_x \left\{ x + \frac{1}{\alpha} E\left[\max(\xi - x, 0)\right] \right\}$$

where $\alpha = 0.1$ and ξ is drawn from a standard normal distribution.

2.5.1.3. *Scalable Farmer.* The original farmer example in [10] has three crops and three scenarios, where a farmer must decide at the beginning of the planting season how much of their land to allocate to each crop. Our tested version is for stress-testing software such as [56]. Instance configuration parameter *numscens* is added to scale up the number of scenarios. By incorporating

this parameter, the scalable instances can create any number of scenarios, so we can draw samples of any size. The problem can be formulated as:

$$
\begin{aligned}
\underset{x,y^\xi,w^\xi}{\text{minimize}} \quad & 150x_1 + 230x_2 + 260x_3 + \mathbb{E}_\xi[(238y_1^\xi + 210y_2^\xi) - (170w_1^\xi + 150w_2^\xi + 36w_3^\xi + 10w_4^\xi)] \\
\text{subject to} \quad & x_1 + x_2 + x_2 \geqslant 500, \\
& \omega_1^\xi x_1 + y_1^\xi - w_1^\xi \geqslant 200, \quad \forall \xi \in \Xi \\
& \omega_2^\xi x_2 + y_2^\xi - w_2^\xi \geqslant 240, \quad \forall \xi \in \Xi \\
& \omega_3^\xi x_3 - w_3^\xi - w_4^\xi \geqslant 0, \quad \forall \xi \in \Xi \\
& w_3^\xi \leqslant 6000, \\
& x, y^\xi, w^\xi \geqslant 0, \quad \forall \xi \in \Xi,
\end{aligned}
$$

where ξ represents the uncertainty associated with future yields and market prices, ω_i^ξ denotes the yield per acre for each crop under scenario ξ, while x_i specifies the number of acres allocated to each crop. Further, y_i^ξ indicates the quantity of crops to be purchased to address any shortfall, and w_i^ξ refers to the quantity of crops to be sold. All scenarios are grouped in threes, with a uniformly distributed pseudorandom number added to the yield values of the original three scenarios.

We further introduced a new feature, denoted as "yield-cv," which represents the coefficient of variation of the crop yields. This inclusion offers the flexibility to introduce variability in problem settings, and is universally applicable to all crops. In cases where it is not explicitly specified, the distribution of the farmer example adheres to the original model with uniform distributions.

2.5.1.4. *Multi Knapsack.* The multi-knapsack problem originates from the stochastic programming problem discussed in [98] and also explored in [52, Chapter 6]. It can be regarded as a multidimensional newsvendor issue that includes substitution effects. In the initial stage, manufacturers make decisions on the product assortment and set inventory levels without precise knowledge of future demand. Subsequently, in the second stage, once the actual demand is revealed, it allocates products for both direct sales and substitutions to maximize profit. In our analysis, we focus on the scenario that involves the sale of six products, each subject to a uniform substitution rate of $\alpha_{ij} = 10\%$. The problem can be formulated as follows:

$$
\begin{aligned}
\text{maximize} \quad & \mathbb{E}_\xi \left[\sum_i -c_i x_i + v_i y_i^\xi + v_i z t_i^\xi + g_i (x_i - (y_i^\xi + z t_i^\xi)) \right] \\
\text{subject to} \quad & y_i^\xi + \sum_{j;j\neq i} z_{ji}^\xi \leqslant d_i^\xi, \\
& z_{ij}^\xi \leqslant \alpha_{ij} (d_j^\xi - y_j^\xi), \\
& z t_i^\xi = \sum_{j;j\neq i} z_{ij}^\xi, \\
& x_i, y_i^\xi, z t_i^\xi, w_i^\xi, z_{ij}^\xi \geqslant 0.
\end{aligned}
$$

Here, x_i represents the quantity of production, while y_i indicates the quantity sold. The variables z_{ij} and zt_i denote the substitution sales of product i using product j and across all other products, respectively. The parameter d_i denotes the demand for each product, c_i is the associated production cost, v_i is the sale price, and g_i reflects the salvage value of unsold products.

2.5.2. Parameter Values. In the interest of reproducibility, Tabe 2.1 shows the parameter values used in experiments except when otherwise noted.

Symbol	Name in SMAIS	NewsVendor	CVaR	Farmer	Knapsack
M	initial_sample_size	20	80	40	80
M_q	assess_size	20	20	40	40
M_r	additional_sample_size	20	20	40	40
K	sp_integral_size	2000	2000	10,000	1,000,000
c_β	adaptive_error_threshold_factor	0.1	0.1	0.1	0.1
N	evaluation_N	500	500	500	500

TABLE 2.1. Parameter Values Used in Experiments.

2.5.3. Method Comparison. For the CVaR problem, we evaluated the performance of three methods: our proposed Adaptive Importance Sampling with Surrogate Modeling (SM-AIS), the regular Monte Carlo (MC) method, and the Markov Chain Monte Carlo Importance Sampling (MCMC-IS) method. Their effectiveness was assessed by measuring the deviation of the estimator for the generated function value from the benchmark and the width of the confidence intervals generated for $\hat{z}$, over various predetermined cutoff times. The benchmark value is achieved by the conventional Monte Carlo method upon convergence. The methods were implemented in serial, without parallelization. We will discuss potential performance improvements for the AIS-SM

method through parallelization in a later section. The experiments were carried out with Python 3.12.1 on a server with dual Intel(R) Xeon(R) Gold 5118 CPUs, 376 GB of RAM, running Ubuntu 22.04.3 LTS.

Due to the necessity for an initial run-time to establish its importance sampling distribution, the MCMC Importance Sampling method does not provide estimations at early cut-off times. Consequently, these instances are denoted by the symbol '-' in the table. When the relative error is less than 0.1%, we consider it to be negligible.

Table 2.2 demonstrates the varied effectiveness of sampling methods in addressing the CVaR problem. Due to the significant disparity between the important region for CVaR estimation and the original distribution p, the regular MC method can be extremely inefficient, and an effective importance sampling distribution is necessary for fast estimation of the function value.

Cut-off Time (Seconds)	MC		MCMC-IS		SM-AIS	
	Error	CI Width	Error	CI Width	Error	CI Width
100	0.22	0.99	-	-	0.013	0.092
200	0.081	0.27	0.015	0.11	negligible	0.024
400	0.050	0.20	0.015	0.05	negligible	0.015

TABLE 2.2. Summary Table for CVaR, with the benchmark function value $\widehat{z} \approx 2.1155$ obtained using numerical integration. The current result record MCMC-IS method runs with 2500 samples for the hastings method. For the proposed SM-AIS method, we start with an initial size of 80 training samples, and assess the surrogate model via error-based estimation. The surrogate model meets the stopping criteria within 6 iterations.

While the MCMC-IS method's slower development of the importance sampling distribution might seem a drawback, it still surpasses the basic MC method due to the usage of the importance function. The SM-AIS method stands out for its rapid creation of an efficient importance sampling distribution, leading to minimal errors and significantly tighter confidence intervals.

We also generated Figure 2.3, 2.4 and 2.5 to represent the convergence of confidence intervals for the estimates obtained by the MC method, the MCMC-IS method, and the SM-AIS method in the estimation of the Conditional Value at Risk (CVaR) over time. The red line represents the trajectory of the integral estimate, and the shaded region encapsulates the 95% confidence interval at different timestamps with different sample sizes. The SM-AIS method demonstrates faster

convergence to the accurate CVaR estimate and yields substantially tighter confidence intervals compared to those produced by other approaches. Notably, with the importance sampling function generated, both the MCMC-IS and SM-AIS methods require far fewer samples compared to the MC method to achieve a stabilized confidence interval.

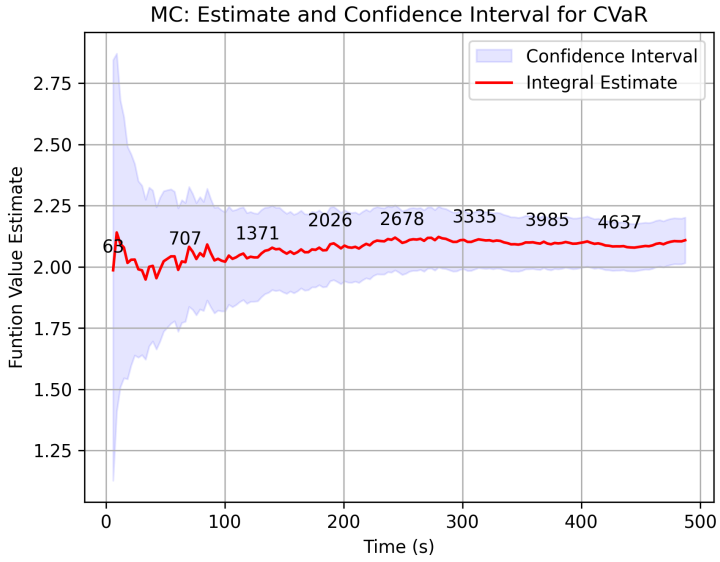

FIGURE 2.3. Convergence of the CVaR estimates over time using the MC method. The red line represents the integral estimate of the CVaR, while the shaded area delineates the 95% confidence interval. Annotations on the plot indicate the sample size used at various time intervals.

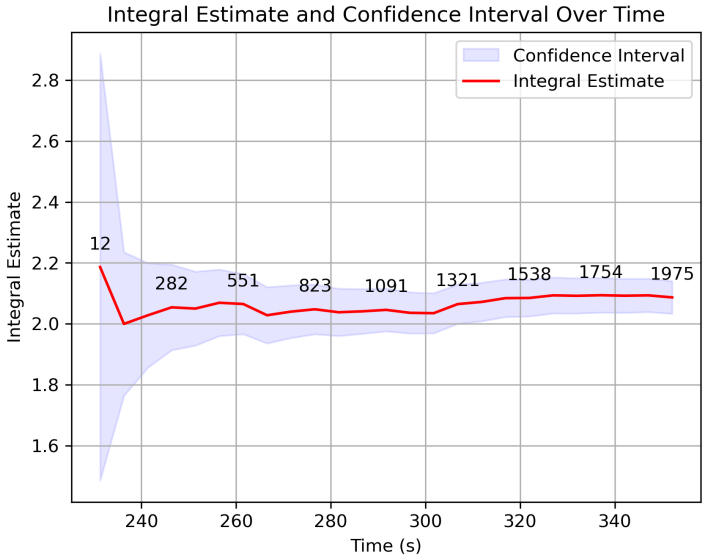

FIGURE 2.4. Convergence of the CVaR estimates over time using the MCMC-IS method. The red line represents the integral estimate of the CVaR, while the shaded area delineates the 95% confidence interval. Annotations on the plot indicate the sample size used at various time intervals.

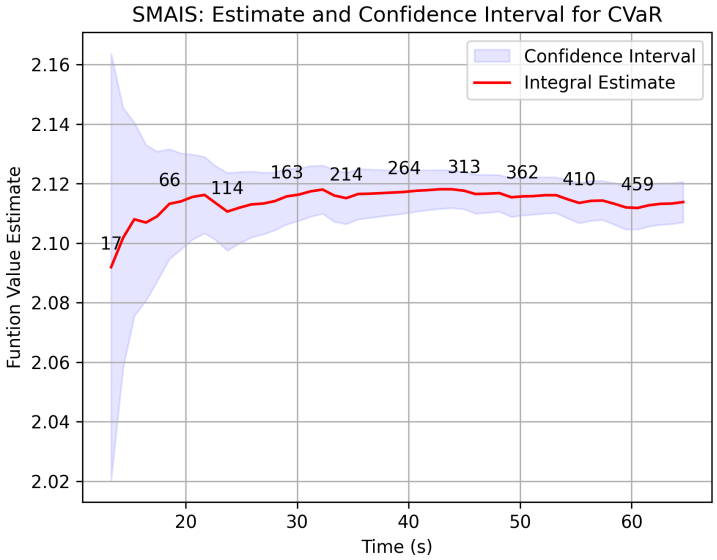

FIGURE 2.5. Convergence of the CVaR estimates over time using the SM-AIS method. The red line represents the integral estimate of the CVaR, while the shaded area delineates the 95% confidence interval. Annotations on the plot indicate the sample size used at various time intervals.

For the rest of the example problems, the efficiency of the MCMC-IS method is constrained by the limited utility of the importance sampling distribution in relatively simple scenarios. Generation of the importance sampling distribution via the MCMC-IS approach is time-consuming and does not offer sufficient evaluation efficiency improvements for certain problems. This challenge is particularly evident for some multi-dimensional optimization problems, where optimally adjusting the MCMC-IS method's parameters proves difficult due to the Kernel Density Package available for use in our Python implementation. Hence, for the rest of the examples, we primarily report the time complexity advantage of our approach over the regular MC method.

As shown in Tables 2.3 and 2.4, in the NewsVendor problem and the farmer problem, the regular Monte Carlo (MC) method achieves efficient convergence, yet the Adaptive Importance Sampling with Surrogate Models (SM-AIS) method further excels by ensuring faster convergence and significantly narrower confidence intervals.

Cut-off Time (Seconds)	MC		SM-AIS	
	Error	CI Width	Error	CI Width
20	2.7	16.7	0.3	0.4
100	1.3	8.2	negligible	0.1
200	0.6	5.7	negligible	< 0.1
400	0.8	4.0	negligible	<0.1

TABLE 2.3. Summary Table for NewsVendor, with the benchmark function value $\widehat{z} \approx 558.6$ obtained using regular MC with $20k$ samples. The current result record MCMC-IS method runs with 1500 samples for the Hastings method. For the proposed SM-AIS method, we start with an initial size of 20 training samples, and assess the surrogate model via error-based estimation. The surrogate model meets the stopping criteria within 2 iterations.

For multidimensional problems such as the multi-knapsack problem, building the importance sampling distribution in Step 3 requires a substantial number of samples, denoted by K, to accurately estimate the integral in the denominator. However, since the integral calculation only involves evaluations of the surrogate model s, the process remains relatively cost-effective, despite the requirement for extensive sampling. Table 2.5 suggests that our method effectively reduces the variance of the final estimator, resulting in a more precise confidence interval. In Section 2.5.4, we demonstrate how the incorporation of parallel computing strategies can markedly decrease the computational time required by high-dimensional integration.

Cut-off Time (Seconds)	MC		SM-AIS	
	Error	CI Width	Error	CI Width
40	2126	4,125	87	418
200	575	1995	negligible	164
400	347	1397	negligible	117
800	104	1018	negligible	83

TABLE 2.4. Summary Table for Farmer, with the actual function value $\widehat{z} \approx 151,600$ obtained with $65k$ samples. The current result records MCMC-IS method run with 2000 samples for hastings method with a fixed step size. For the proposed Surrogate-IS method, we start with an initial size of 20 training samples, and assess the surrogate model via error-based estimation. The surrogate model meets the stopping criteria within 2 iterations.

Cut-off Time (Seconds)	MC		SM-AIS	
	Error	CI Width	Error	CI Width
500	220	1,198	117	433
1000	53	862	41	121
1500	37	713	negligible	95

TABLE 2.5. Summary Table for Multi-Knapsack, with the actual function value $\widehat{z} \approx 24,860$ obtained with $20k$ samples. The current result records MCMC-IS method run with 1000 samples for hastings method with a fixed step size. For the proposed Surrogate-IS method, we start with an initial size of 80 training samples, and assess the surrogate model via error-based estimation. The surrogate model meets the stopping criteria within 2 iterations.

2.5.4. Parallel Speed Up. The SM-AIS method offers the advantage of straightforward parallelization, leading to considerable performance enhancements. To demonstrate, we present a comparison between the parallelized and non-parallelized implementations of the SM-AIS method when applied to the multi-knapsack problem. For this purpose, we employ a fixed set of parameters and evaluate the run-time performance of the SM-AIS method across a variety of process counts.

Our implementation of parallelization was focused on three critical phases of SM-AIS: In Step 3, parallel processing was employed to evaluate the denominator when constructing the importance sampling distribution q_s^k; Step 4 leveraged parallelism for performing rejection sampling for evaluating the surrogate model's quality through error-based assessment; and in Step 6, parallel processes were utilized once more for drawing samples from q_s^k for the final estimation. Further discussion of parallel implementation can be found in Section 2.4.4. In our experiments, the batch sizes were set to $B_a = 200$ for the quality assessment phase and $B_e = 100$ for the final estimation phase.

Number of Processes	Runtime (seconds)	Speed Up Factor
1	1037.278	1
2	524.380	1.978
4	289.270	3.587
8	155.382	6.675
12	107.249	9.672
16	89.374	11.609

TABLE 2.6. Runtime performance of the SM-AIS method for the multi-knapsack problem using different numbers of processes. We used $K = 1000,000$ to evaluate the denominator to construct the importance sampling distribution q_s^k, and used $N = 200$ samples for the final estimation.

Table 2.6 outlines the runtime of the SM-AIS method under varying numbers of parallel processes, and Figure 2.6 visually represents the scaling efficiency, both highlighting the effectiveness of parallelization in reducing the runtime.

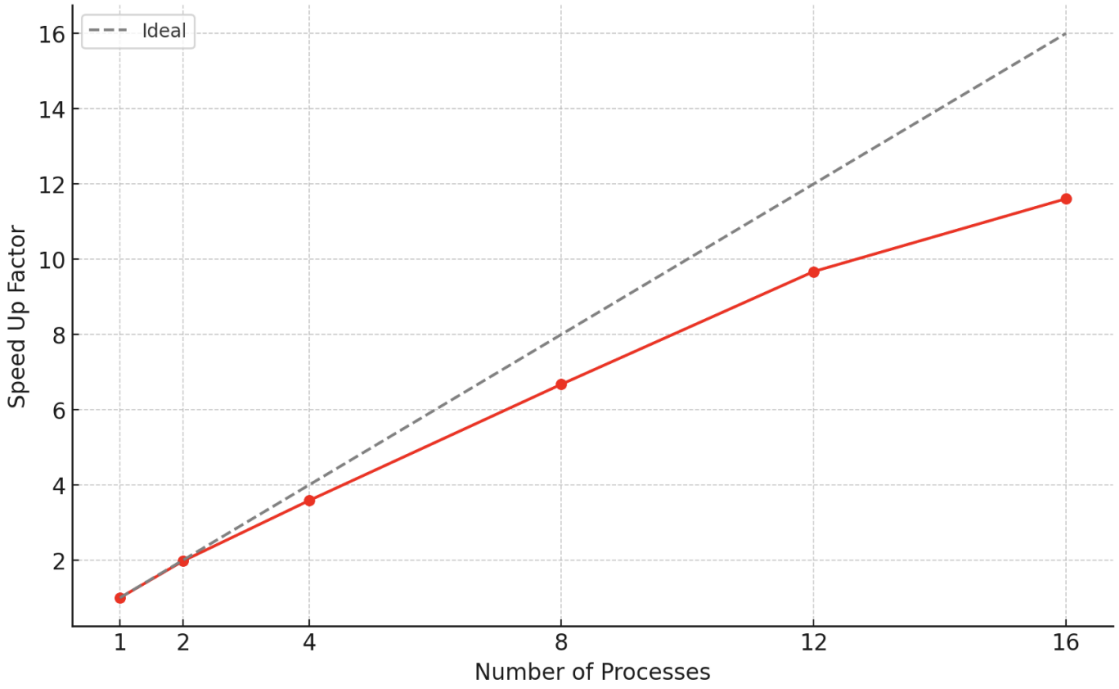

FIGURE 2.6. The Speed Up Factor for SM-AIS with Increasing Parallel Processes for multi-knapsack example. The red line indicates the actual speed-up of the algorithm as the number of processes increases, while the gray dashed line represents the ideal linear speed-up.

2.5.5. Parameter Selection and Stability. In this section we report the result on various parameter selections and how they affect the run-time and the accuracy of the final estimation for

the SM-AIS method.

2.5.5.1. *Sample Size K for importance sampling distribution scaling.* In practice, the parameter K can be determined by verifying that the Monte Carlo integral of the probability density function p with K samples is close to 1. The choice of sample size K is particularly important in Step 3, where it affects the evaluation of the integral of the surrogate model s over its domain. This step determines the appropriate scaling for the importance sampling distribution introduced by the surrogate model s. The size of K must be sufficiently large to ensure that the integral estimation is reliable.

K	Error	CI Width	MC integral for p
1000	negligible	0.22	1.000
100	2.14	0.23	0.995
10	82.40	0.19	0.873

TABLE 2.7. Influence of Sample Size K on Estimation Accuracy in the NewsVendor Example, with the actual function value $\widehat{z} \approx 558.6$. Here, $M = 20$ samples were used to construct the surrogate model employing kriging. The error in the final estimation was calculated using $N = 100$ samples. The surrogate model satisfied the stopping criteria within 1 iteration.

Table 2.7 represents how varying parameter K in the NewsVendor Problem affects the bias in the final estimations. In addition, it includes the results of the Monte Carlo integration p, which serves as an empirical benchmark for determining K's adequacy. The results reveal that while the SM-AIS method still converges with an inadequately small K, it results in biased outcomes. In contrast, achieving a Monte Carlo integral of p close to 1 indicates a minimal error in the final estimation, thereby serving as a guide to select an appropriate K.

2.5.5.2. *Scaling factor c_β for choosing additional training sample.* The SM-AIS approach exhibits relative robustness with respect to the scaling factor c_β used to determine the stopping criteria and to select additional training samples. Its impact on overall performance remains minimal, provided that c_β is maintained within a practical range. Taking into account the problem's sensitivity and user preferences, a c_β set between 5% and 20% has proven to be both efficient and effective based on practical experience. For the majority of our experiments, except those focused

on parameter sensitivity analysis, we used $c_\beta = 10\%$.

2.5.5.3. *Choice of Surrogate Model Type.* When selecting a surrogate model type, utilizing prior knowledge from the application domain can certainly be advantageous, potentially improving the surrogate model's performance. This knowledge helps to choose a model type that is better suited to grasp the essential features of the function or system being modeled. However, for the problems we have experimented with, the surrogate model type's influence on performance is not significant, provided that an ample number of samples are available for training. This suggests that the adequacy of the training data plays a more critical role than the specific choice of model type in determining the effectiveness of the SM-AIS method.

2.5.5.4. *Initial Sample Size and Assessment Sample size.* Due to the adaptive framework of the SM-AIS method, the choice of the initial sample size M for constructing the initial surrogate model is relatively flexible, as long as an adequate number M_r of uniform random samples is used to evaluate the quality of intermediate surrogate models and guide the decision on stopping criteria. However, choosing a very small M may lead to a poor initial model, potentially causing detours in the refinement process and increasing the iterations needed to meet the stopping criteria, thus increasing the total evaluations of the function g required for model construction.

The sample size for assessing intermediate surrogate models is similarly adaptable, with the caveat that it should not be too small to overlook significant errors in a deficient model or too large to avoid wasting computational resources evaluating the function g. Nevertheless the number of samples M_q drawn from the importance sampling distribution q_s should ideally match or be less than those used for constructing the model, as over-sampling from an underdeveloped model offers little to no benefit for its enhancement.

In essence, the initial and assessment sample sizes for the SM-AIS method are adaptable, but should be kept above a minimum to prevent early stopping and excessive sampling in the final phase.

As an example, we examine the impact of varying initial sample sizes on the surrogate model construction process and its subsequent effect on final estimations, using the farmer example. We maintained consistency with the values shown in Table 2.1, with the exception of the initial

sample size M. Additionally, the number of samples, M_q, drawn from q_s to evaluate the surrogate model s, was set at $\min(20, 2M)$. Finally, to ensure an unbiased final estimation, we incorporated $M_r = 40$ additional uniformly distributed random samples for evaluation in each iteration. The total number of function g evaluations required for constructing the surrogate model, in this case, is $M + k(M_q + M_r)$, where k represents the iterations needed to meet the stopping criteria for the construction of the surrogate model. We also provide runtime for constructing the surrogate model.

For the final evaluation, we set a fixed sample size of $N = 200$ and reported the associated error and confidence interval based on these $N = 200$ samples. Table 2.8 summarizes the result. As observed in the table, the accuracy of the final estimation remains largely unaffected by variations in initial or assessment sample sizes, compared to the scale of the actual function value. However, smaller initial sample sizes may require a greater number of iterations (k) to develop an acceptable surrogate model, which in turn results in increased runtime.

M	40	20	8	3
k	1	1	2	3
Total g evaluation	100	80	120	141
RunTime(s)	10.88	8.43	12.66	15.44
Error	negligible	negligible	negligible	negligible
CI Width	839	650	1103	1543

TABLE 2.8. Influence of Initial Sample Size M on Estimation Efficiency in the Farmer Example, with the actual function value $\widehat{z} \approx 151,600$. We draw $\min(20, 2M)$ samples from the intermediate surrogate models for error-based assessment, in addition to a fixed 40 uniformly random assessment samples. Error and CI width obtained with $N = 200$ final estimation samples.

CHAPTER 3

Optimality Gap Estimation for Multi-Stage Stochastic Programming

3.1. Introduction

Another important aspect to consider for a candidate solution is its optimality gap. For a general optimization problem formulated as

$$z^* = \min_x E_\xi[g(x, \xi)],$$

the optimality gap measures the deviation of the objective function value from the optimal, defined by:

$$\mathcal{G}_{\widehat{x}} = E_\xi[g(\widehat{x}, \xi)] - \min_x E_\xi[g(x, \xi)]. \tag{3.1}$$

A smaller optimality gap indicates a high-quality solution.

In this chapter, we focus on constructing confidence intervals for the optimality gap in multi-stage stochastic programs (MSSP), following the work in [18]. These intervals are useful for subsequent analysis and for making decisions about the allocation of computing resources.

3.1.1. Background. A multistage stochastic program is an optimization problem that extends two-stage stochastic programming to accommodate multiple stages of sequential decision-making under uncertainty. At each stage, decision-makers respond based on the realized uncertainties up to that point, with each decision influencing the subsequent choices and associated uncertainties. This iterative decision-making process helps capture the complexity of real-world systems where each sequential decision can significantly impact future outcomes. Originally proposed by [19], it has been applied to a wide spectrum of areas, including energy planning [79], hydrothermal scheduling [24], vehicle routing [50], financial modeling [105], supply chain management [51], and many others.

Mak et al. [67] developed sampling-based upper and lower bound estimators to construct a confidence interval on the optimality gap for two-stage stochastic programming based on asymptotic normality. Additional computational results can be found in, for example, [64]. Shapiro [93] discussed the difficulty in extending such sampling methods to multistage linear programming problems and proposed a conditional sampling procedure for constructing consistent lower bounds. Chiralaksanakul and Morton [18] further proposed two Monte Carlo sampling methods to construct upper bounds for the optimality gap in the case of stage-wise independence and stage-wise 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.

A few papers discussed how to assess the solution quality for multi-stage programming with the additional assumption of inter-stage independence. These algorithms are mostly adaptations of the SDDP algorithms [79]. Homem and Godinho [47] presented an alternative stopping criterion for SDDP that is more statistically robust. Kozmik and Morton [58] showed that importance sampling could be used to improve upper bound estimators in the risk-averse setting. Their work improves the results of [95]. De et al. [24] proposed an algorithm that can construct a confidence interval of the optimality gap using a single scenario tree.

In the following sections, we explore a Multiple Replication Procedure (MRP) designed to assess the solution quality in multistage stochastic programming without the inter-stage independence assumption, based on the work [18, 67, 93]. In addition, we discuss possible adaptations of Sequential Sampling techniques that estimate the optimality gap for multistage stochastic programming for a sequence of candidate solutions, under an increased sample size. The sequential sampling procedures were first proposed in [6, 7, 8] for two-stage stochastic programs.

A software implementation of both the MRP method and the Sequential Sampling method for assessing the solution quality for MSSP is available as part of the mpi-sppy package [56], which is available for download from `https://github.com/Pyomo/mpi-sppy`

3.1.2. Multistage Stochastic Program Formulation. The dynamic realization of uncertainty in an MSSP is typically modeled as a stochastic process $\{\boldsymbol{\xi}_t\}_{t=2}^{T}$, where the realized values $\boldsymbol{\xi}_t = \xi_t$ are revealed at the end of stage $t-1$. Therefore, we refer to $\boldsymbol{\xi}_t$ and ξ_t only for $t = 2, \ldots, T$.

The realized values of the stochastic process up to and including stage t are denoted as

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

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

At each stage t, a decision x_t is made with only the knowledge of $\vec{x}^{t-1}$ and $\vec{\xi}^t$, and is independent of future decisions and realizations of $\boldsymbol{\xi}_{t+1}, \ldots, \boldsymbol{\xi}_T$. This independence is known as non-anticipativity. We use the notation $\vec{x}^t$ for $1 \leqslant t \leqslant T$ to represent the decisions for all stages up to, and including, stage t, i.e.,

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

The multi-stage stochastic problem (MSSP) can be formulated recursively as

$$\begin{aligned} z^* = \min_{x_1} \quad & \mathbb{E}_{\boldsymbol{\xi}_2}[\phi_2(x_1; \boldsymbol{\xi}_2)], \\ s.t. \quad & x_1 \in X_1, \end{aligned} \tag{3.2}$$

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

$$\begin{aligned} \phi_t(\vec{x}^{t-1}; \vec{\xi}^t) \quad = \quad & \min_{x_t} \mathbb{E}_{\vec{\boldsymbol{\xi}}^{t+1}}[\phi_{t+1}((\vec{x}^{t-1}, x_t); \vec{\boldsymbol{\xi}}^{t+1}) | \vec{\boldsymbol{\xi}}^t = \vec{\xi}^t], \\ s.t. \quad & x_t \in X_t\left(\vec{x}^{t-1}, \vec{\xi}^t\right), \end{aligned} \tag{3.3}$$

and the final stage ($t = T$) optimizes the problem-specific objective function ϕ'_T, given all prior decisions and the complete scenario $\vec{\xi}^T$.

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

The feasible region constraints at each stage can be easily incorporated into the objective function via extended representation,

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

This way, the MSSP problem (3.2) can be simplified to

$$z^* = \min_{x_1} \mathbb{E}_{\boldsymbol{\xi}_2}[g_2(x_1; \boldsymbol{\xi}_2)], \tag{3.4}$$

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

$$g_t(\vec{x}^{t-1}; \vec{\xi}^t) = \min_{x_t} \mathbb{E}_{\vec{\boldsymbol{\xi}}^{t+1}}[g_{t+1}((\vec{x}^{t-1}, x_t); \vec{\boldsymbol{\xi}}^{t+1}) | \vec{\boldsymbol{\xi}}^t = \vec{\xi}^t], \tag{3.5}$$

and

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

for some problem-specific objective function value.

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.

3.2. Prerequisites

In this section, we discuss several procedures that will be later used in constructing a confidence interval around the optimality gap for MSSP.

For multi-stage problems, it is important to effectively model the uncertainty across multiple decision-making stages. One approach to achieve this is through the construction of a scenario tree. In a scenario tree, each node represents a specific state or decision point within the model, and branches represent potential transitions between states, each associated with a specific probability. The paths from the root to the leaves represent complete scenarios.

At each stage t, the scenarios that share the same realization up to that point converge at a common node. 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, but merely a structural placeholder. For any node $\vec{\xi}^t$, we use $\Gamma(\vec{\xi}^t)$ to denote a general subtree rooted at $\vec{\xi}^t$. Depending on the context, $\Gamma(\vec{\xi}^t)$ can be generated by either incorporating the entire sequence of decisions and outcomes from the original scenario tree, or by selectively sampling paths. We will discuss 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.

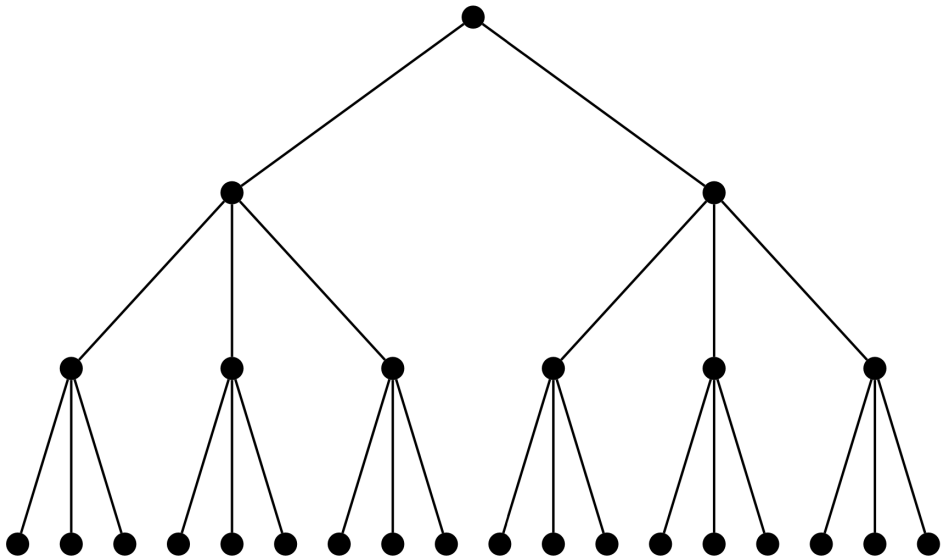

FIGURE 3.1. Example of scenario tree for a 4-stage stochastic programming problem.

The cardinality of the 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 $t-1$.

3.2.1. Conditional Sampling. For the simplified two-stage SP problem

$$z^* = \min_{x_1} \mathbb{E}_{\boldsymbol{\xi}_2}[g_2(x_1; \boldsymbol{\xi}_2)],$$

the Monte Carlo approximation for a two-stage stochastic program is formulated as follows:

$$\min_{x_1} \frac{1}{N} \sum_i g_2\left(x_1; \xi_2^i\right), \tag{3.6}$$

Here, ξ_2^i are i.i.d. realizations of the random variable $\boldsymbol{\xi}_2$. Due to the complex structure of the scenario tree in multistage contexts, this Monte Carlo approximation for the two-stage SP problem cannot be applied directly to multistage settings.

Following the work of [18, 93], we describe a conditional sampling procedure that constructs a subtree $\Gamma(\vec{\xi}^t)$ given any node $\vec{\xi}^t$ in the original scenario tree. This mimics the Monte Carlo approach for the two-stage SP within a multistage framework.

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

$t \geqslant 2$. Here, n_{t+1} is the sample size. In general the sample size associated with each node does not 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, that is, $t = T$. The total number of 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 in 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,

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

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.

Algorithm 1 contains the conditional sampling procedure.

Algorithm 1 Conditional Sampling Procedure for Constructing Subtrees

Input: stage t, node $\vec{\xi}^t$, sequence of sample sizes $\{n_\tau\}_{\tau=t+1}^T$
Output: A subtree rooted at $\vec{\xi}^t$
Initialize the subtree $\Gamma(\vec{\xi}^t)$.
for $\tau = t+1$ **to** T **do**
    for all nodes $\vec{\xi}^{\tau-1,i}$ in stage $\tau - 1$ in $\Gamma(\vec{\xi}^t)$ **do**
        Sample n_τ observations $(\xi_{\tau,1}, \ldots, \xi_{\tau,n_\tau})$ from $F(\xi_\tau|\vec{\xi}^{\tau-1,i})$
        for $j = 1$ **to** n_τ **do**
            Construct descendant node $\vec{\xi}^{\tau,j} = (\vec{\xi}^{\tau-1,i}, \xi_{\tau,j})$
            Add $\vec{\xi}^{\tau,j}$ to $\Gamma(\vec{\xi}^t)$
return $\Gamma(\vec{\xi}^t)$.

3.2.2. Optimization given a scenario tree. Given a finite scenario tree $\Gamma(\vec{\xi}^1)$, we assume that we can find an optimal solution for the following optimization problem:

$$\widehat{z}^* = \min_{x_1} \frac{1}{n_2} \sum_{\xi_{2,i} \in D(\vec{\xi}^1)} \widehat{g}_2(x_1; \xi_{2,i}), \tag{3.7}$$

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

$$\widehat{g}_t(\vec{x}^{t-1}; \vec{\xi}^{t,i}) = \min_{x_t} \frac{1}{n_{t+1}} \sum_{\xi_{t+1,j} \in D(\vec{\xi}^{t,i})} \widehat{g}_{t+1}(\vec{x}^t; (\vec{\xi}^{t,i}, \xi_{t+1,j})), \tag{3.8}$$

and the objective function in the final stage $\widehat{g}_T(\vec{x}^{T-1}, \vec{\xi}^{T,i})$ is consistent with the original MSSP formulation as specified in (3.2).

When the original problem is represented by the finite scenario tree $\Gamma(\vec{\xi}^1)$, the optimal value derived from (3.7) aligns with that of (3.4). If the scenario tree $\Gamma(\vec{\xi}^1)$ is constructed using conditional sampling as detailed in Algorithm 1, then the optimization problem expressed in (3.7) serves as a Monte Carlo approximation to the original MSSP problem defined in (3.2).

3.2.3. Finding a Feasible Policy. In MSSP, a feasible solution is specified by a sequence of decisions $\{\widehat{x}_i\}_{i=1}^T$ that satisfy all constraints at different stages in varying scenarios. While in two-stage cases with complete recourse, it is relatively straightforward to find a feasible solution that satisfies $x_2 \in X_2\left(x_1, \vec{\xi}^2\right)$, this process is less trivial in a multistage context.

Suppose that a decision $\widehat{x}_1$ for the first stage is obtained by solving an approximation of the MSSP problem as given in (3.2). Given a specific scenario $\vec{\xi}^T$, we are interested in identifying an associated feasible decision policy $\{\widehat{x}_i\}_{i=1}^T$ for different stages. This policy should be non-anticipative, with each decision $\widehat{x}_i$ meeting the stage-wise constraints.

To construct a feasible decision policy for $\vec{\xi}^T$, we simulate the process of finding a solution for the original problem, but this time in a roll out manner. Starting from stage two and proceeding to stage T, at each stage t. we fix the decisions made up to that point, $\{\widehat{x}_i\}_{i=1}^{t-1}$. We then generate a subtree $\Gamma(\vec{\xi}^t)$ rooted at the current scenario $\vec{\xi}^t$ using the conditional sampling method as in Algorithm 1. The decision $\widehat{x}_t$ at each stage t is then determined by solving the subsequent optimization problem,

$$\widehat{x}_t = \operatorname*{argmin}_{x_t} \frac{1}{n_{t+1}} \sum_{\xi_{t+1,i} \in D(\vec{\xi}^t)} \widehat{g}_{t+1}((\vec{\widehat{x}}^{t-1}, x_t); (\vec{\xi}^t, \xi_{t+1,i})), \tag{3.9}$$

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

$$\widehat{g}_\tau(\vec{x}^{\tau-1}; \vec{\xi}^{\tau,i}) = \min_{x_\tau} \frac{1}{n_{\tau+1}} \sum_{\xi_{\tau+1,j} \in D(\vec{\xi}^{\tau,i})} \widehat{g}_{\tau+1}((\vec{x}^{\tau-1}, x_\tau); (\vec{\xi}^{\tau,i}, \xi_{\tau+1,j})), \tag{3.10}$$

and $\widehat{g}_T(\vec{x}^{T-1}; \vec{\xi}^{T,i})$ mirrors the objective function for the final stage of the original MSSP problem. 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 3.2.2. Indeed, after we sample the subtree from a node, that subtree essentially defines a new MSSP problem. The process is summarized as in Algorithm 2.

Algorithm 2 Generating a Feasible Solution

1: **Input**: First stage solution $\widehat{x}_1$, scenario $\vec{\xi}^T$
2: **Output**: Feasible decision policy $\{\widehat{x}_i\}_{i=1}^T$
3: **for** $t = 2$ **to** T **do**
4: Independently sample a new subtree $\Gamma(\vec{\xi}^t)$ with root $\vec{\xi}^t$ using Algorithm 1.
5: Solve the associated optimization problem (3.9).
6: Record the optimal value as $\widehat{x}_t$.
7: **return** $\{\widehat{x}_i\}_{i=1}^T$

3.3. MRP Estimators for the Optimality Gap

We begin by discussing the MRP estimators for two-stage stochastic programming problems, then explore the challenges in directly extending these estimators to multi-stage stochastic programming. Finally, we consider MRP estimators designed for multi-stage stochastic problems, which are developed in a manner similar to those for two-stage problems.

3.3.1. MRP Estimators for Two-Stage SP. Considering a general form of a two-stage SP problem

$$z^* = \min_{x_1} \mathbb{E}_{\boldsymbol{\xi}_2}[g_2(x_1; \boldsymbol{\xi}_2)].$$

Given a candidate first stage solution $\widehat{x}$, one way to construct a point estimator for the optimality gap $\mathcal{G}_{\widehat{x}_1}$ is to sample N i.i.d. observations $\xi_2^1, \xi_2^2, \ldots, \xi_2^N$ for $\boldsymbol{\xi}_2$, and then compute the associated gap

$$\widehat{\mathcal{G}}_N(\widehat{x}) = \frac{1}{N}\sum_{i=1}^{N} g_2\left(\widehat{x}; \xi_2^i\right) - \min_{x_1} \frac{1}{N}\sum_{i=1}^{N} g_2\left(x_1; \xi_2^i\right) \tag{3.11}$$

Considering the expected value, we have

$$\begin{aligned}
\mathbb{E}[\widehat{\mathcal{G}}_N(\widehat{x})] &= \frac{1}{N}\sum_{i=1}^{N}\mathbb{E}\left[g_2\left(\widehat{x};\boldsymbol{\xi}_2^i\right)\right] - \mathbb{E}\left[\min_{x_1}\frac{1}{N}\sum_{i=1}^{N} g_2\left(x_1;\boldsymbol{\xi}_2^i\right)\right] \\
&\geqslant \mathbb{E}\left[g_2\left(\widehat{x};\boldsymbol{\xi}_2\right)\right] - \min_{x_1}\mathbb{E}\left[\frac{1}{N}\sum_{i}^{N} g_2\left(x_1;\boldsymbol{\xi}_2^i\right)\right] \\
&= \mathbb{E}\left[g_2\left(\widehat{x};\boldsymbol{\xi}_2\right)\right] - \min_{x_1}\mathbb{E}\left[g_2\left(x_1;\boldsymbol{\xi}_2\right)\right] \\
&= \mathbb{E}\left[g_2\left(\widehat{x};\boldsymbol{\xi}_2\right)\right] - z^*,
\end{aligned}$$

thus $\widehat{\mathcal{G}}_N(\widehat{x})$ is a conservative estimator that tends to overestimate the optimality gap. As the construction of $\widehat{\mathcal{G}}_N(\widehat{x})$ involves optimization over a sample mean, traditional statistical analysis for constructing confidence intervals does not directly apply. Nonetheless, this limitation can be addressed by employing the batch means method. This statistical approach is commonly used in Monte Carlo simulation to manage the correlations between simulation data. By segmenting the Monte Carlo simulations into multiple batches, each consisting of independent sets of observations, we can treat each batch's output as an individual realization of the estimator. In estimating the optimality gap for two-stage stochastic programming, we generate multiple independent estimators $\widehat{\mathcal{G}}_N^k(\widehat{x})$ across N_g batches of observations $\xi_2^{k,1}, \xi_2^{k,2}, \ldots, \xi_2^{k,N}$, for $k = 1, \ldots, N_g$. As these estimators are independent and identically distributed, we can employ the Central Limit Theorem to approximate a one-sided confidence interval for the optimality gap Specifically, the average of these estimators is calculated as:

$$\bar{\mathcal{G}} = \frac{1}{N_g}\sum_{k}\widehat{\mathcal{G}}_N^k(\widehat{x}),$$

and their sample variance is:

$$s_g^2 = \frac{1}{N_g - 1}\sum_{k}(\widehat{\mathcal{G}}_N^k(\widehat{x}) - \bar{\mathcal{G}})^2.$$

The approximate $(1-\alpha)$-level confidence interval is then determined as:

$$[0, \bar{\mathcal{G}} + \frac{t_{N_g-1,\alpha}s_g}{\sqrt{N_g}}], \tag{3.12}$$

here $t_{N_g-1,\alpha}$ is the critical value from the student-t distribution.

The constructed point estimator $\bar{\mathcal{G}}$ and the resulting confidence interval are referred to as the *Multiple Replication Procedure (MRP)* estimators, as they involve multiple replications. This approach was first introduced by [67]. In situations where the computation of the MRP estimators

becomes costly for complex problems, [5] discusses the usage of a single or two replications to obtain the estimators.

When only one replication is used, that is, when only one batch of scenarios is used, the point estimator $\bar{G}$ aligns with the expression in (3.11). Although in this scenario we cannot compute the sample variance of the point estimators due to having only one estimator, by introducing $\widehat{x}_N$ as the optimal solution to the sampling problem $\min_{x_1} \frac{1}{N}\sum_{i=1}^{N} g_2\left(x_1;\xi_2^i\right)$ and writing $\widehat{G}_N(\widehat{x})$ in (3.11) as

$$\widehat{G}_N(\widehat{x}) = \frac{1}{N}\sum_{i=1}^{N}[g_2\left(\widehat{x};\xi_2^i\right) - g_2\left(\widehat{x}_N;\xi_2^i\right)],$$

we can obtain a sample variance estimator as

$$s_g^2 = \frac{1}{N-1}\sum_{i}\left[\left(g_2\left(\widehat{x};\xi_2^i\right) - g_2\left(\widehat{x}_N;\xi_2^i\right)\right) - \widehat{G}_N(\widehat{x})\right]^2 \tag{3.13}$$

Due to the dependency between the optimal solution $\widehat{x}_N$ and the scenario $\{\xi_2^i\}$, the point estimator $\widehat{G}_N(\widehat{x})$ can no longer be considered as a sample mean of independent samples. Nonetheless, it is still possible to construct a confidence interval using one replication, following the same format as in (3.12). The asymptotic validity of this confidence interval was established under certain conditions in [5]

In practice, an estimator produced by a single batch of samples may not be accurate. Therefore, [5] suggested using two replications to generate estimators by averaging the point estimators and the sample variance estimators generated by the two batches, so that

$$\bar{G}' = \frac{1}{2}\left(\widehat{G}_N^1 + \widehat{G}_N^2\right),$$

and

$$s_g^{2\prime} = \frac{1}{2}\left(s_{g,1}^2 + s_{g,2}^2\right),$$

where $\widehat{G}_N^{\cdot}$ and $s_{g,\cdot}^2$ are computed as in (3.11) and (3.13).

These estimators using one replication and two replications are referred to as *Single Replication Procedure (SRP)* and *Average 2 Replication Procedure (A2RP)* estimators. We refer readers to [5] for more details of the SRP and A2RP results.

3.3.2. MRP Estimators for Multi-Stage SP. The MRP estimators for the two-stage SP do not extend directly to MSSP. The complication arises from the requirement to minimize the average of $\frac{1}{N}\sum_i g_2(x_1;\boldsymbol{\xi}_2^i)$ across samples, where, in a multi-stage context, each function g_2 may exhibit considerable complexity.

In the following we discuss the MRP procedure for constructing a confidence interval around the optimality gap for MSSP following [18], by considering an upper bound on the function value associated with a given first-stage solution $\widehat{x}_1$, and a lower bound on the optimal value. Note that with our notation, the function value for a given first stage solution $\widehat{x}_1$ is

$$\mathbb{E}_{\boldsymbol{\xi}_2}[g_2(\widehat{x}_1;\boldsymbol{\xi}_2)],$$

and the function value associated with a complete decision policy $\vec{x}^T$ under a given scenario $\vec{\xi}^T$ is $g'_T(\vec{x}^T;\vec{\xi}^T)$.

3.3.2.1. *Upper Bound* . We start by identifying an upper bound for the function 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 2. It is easy to see that $\mathbb{E}[g'_T(\tilde{x}(\widehat{x}_1,\vec{\boldsymbol{\xi}}^T);\vec{\boldsymbol{\xi}}^T)]$ is an upper bound for the function value at $\widehat{x}_1$, as it is feasible (assuming relatively complete recourse), but not necessarily optimal. Therefore, we have

$$\mathbb{E}[g'_T(\tilde{x}(\widehat{x}_1,\vec{\boldsymbol{\xi}}^T);\vec{\boldsymbol{\xi}}^T)] \geqslant \mathbb{E}_{\boldsymbol{\xi}_2}[g_2(\widehat{x}_1;\boldsymbol{\xi}_2)]$$

While it may not be possible to compute $\mathbb{E}[g'_T(\tilde{x}(\widehat{x}_1,\vec{\boldsymbol{\xi}}^T);\vec{\boldsymbol{\xi}}^T)]$ 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

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

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 $\mathbb{E}g'_T(\tilde{x}(\widehat{x}_1,\vec{\boldsymbol{\xi}}^T);\vec{\boldsymbol{\xi}}^T)$:

$$\mathbb{E}_{\boldsymbol{\xi}_2} g_2(\widehat{x}_1;\boldsymbol{\xi}_2) \leqslant \mathbb{E}g'_T(\tilde{x}(\widehat{x}_1,\vec{\boldsymbol{\xi}}^T);\vec{\boldsymbol{\xi}}^T) \leqslant U_n + \frac{t_{n_u-1,\alpha}s_u}{\sqrt{n_u}} \qquad w.p. \approx 1-\alpha, \tag{3.14}$$

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 $\mathbb{E}g'_T(\tilde{x}(\hat{x}1, \vec{\xi}^T), \vec{\xi}^T)$ by generating n_u independent sampled scenario trees $\Gamma(\vec{\xi}^{1,i})$ using the conditional sampling method described in Section 3.2.1. Each scenario tree can be regarded as a batch of scenarios, as each leaf node represents a scenario. Although within each $\Gamma(\vec{\xi}^{1,i})$ the scenarios are not i.i.d., we can still find an estimator using the batch means method described above. Consider each tree $\Gamma(\vec{\xi}^{1,i})$, for the N_i leaf nodes on $\Gamma(\vec{\xi}^{1,i})$, we find a feasible solution for each and compute the corresponding function values as $\{w_{i,j}\}_{j=1}^{N_i}$. Aggregating the function values from the leaf nodes of $\Gamma(\vec{\xi}^{1,i})$ provides us with a point estimator:

$$W_i = \frac{1}{N_i} \sum_j w_{i,j} \tag{3.15}$$

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

Now, let U_n represent the sample mean for W_i and s_u^2 the standard sample variance. Given that the scenario trees are generated in an independent and identically distributed (i.i.d.) manner, we can utilize the Central Limit Theorem (CLT) to establish an approximate one-sided confidence interval for $\mathbb{E}g'_T(\tilde{x}(\hat{x}_1, \vec{\boldsymbol{\xi}}^T); \vec{\boldsymbol{\xi}}^T)$ using the same expression as (3.14):

$$\mathbb{E}_{\boldsymbol{\xi}_2} g_2(\hat{x}_1; \boldsymbol{\xi}_2) \leqslant \mathbb{E}g'_T(\tilde{x}(\hat{x}_1, \vec{\boldsymbol{\xi}}^T); \vec{\boldsymbol{\xi}}^T) \leqslant U_n + \frac{t_{n_u-1,\alpha} s_u}{\sqrt{n_u}} \qquad w.p. \approx 1-\alpha, \tag{3.16}$$

3.3.2.2. *Lower Bound* . In a two-stage SP problem, by Jensen's Inequality, it is evident that

$$\begin{aligned} \mathbb{E}[\min_{x \in X} \frac{1}{n} \sum_{i=1}^{n} g(x; \xi_i)] &\leqslant \min_{x \in X} \mathbb{E}[\frac{1}{n} \sum_{i=1}^{n} g(x; \xi_i)] \\ &= \min_{x \in X} \mathbb{E}[g(x, \boldsymbol{\xi})], \end{aligned}$$

The same conclusion is applicable when we extend it to multistage stochastic programming. To be exact, let z^* be the optimal value for (3.2) and $\hat{z}^*$ be the optimal value for (3.7). Then we have

$$\mathbb{E}[\hat{z}^*] \leqslant z^*$$

This result can be established by applying Jensen's inequality iteratively at each stage of the decision-making process. For a detailed proof of this result, we refer readers to [18].

As in the previous case, directly computing $\mathbb{E}[\widehat{z}^*]$ is generally not tractable. To find an estimator for $\mathbb{E}[\widehat{z}^*]$, we sample n_l independent scenario trees using the conditional sampling method described in Section 3.2.1, and solve for the optimal value of the problem described in (3.7). In practice, for certain types of problems, it may only be possible to compute a lower bound on the optimal value. Denote the lower bounds as $(\ell_1, \ldots, \ell_{n_l})$. By computing the sample mean L_n and the standard sample variance s_l^2 of these lower bounds, we can construct an approximate one-sided confidence interval as follows:

$$z^* \geqslant \mathbb{E}\widehat{z}^* \geqslant L_n - \frac{t_{n_l-1,\alpha} s_l}{\sqrt{n_l}} \qquad w.p. \approx 1-\alpha. \tag{3.17}$$

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

3.3.2.3. *Confidence interval.* The confidence interval for the optimality gap in multistage stochastic programming (MSSP) can be constructed by combining the upper bound obtained in Section 3.3.2.1 with the lower bound from Section 3.3.2.2. Utilizing the Boole–Bonferroni inequality, equations (3.16) and (3.17) together yield the following approximate $(1-2\alpha)$-level confidence interval for the optimality gap at $\widehat{x}_1$:

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

The construction of the upper bound involves sampling n_u scenario trees, while the lower bound requires sampling n_l scenario trees. Although it is necessary for the trees within each bound calculation to be independent, there is no requirement for independence across the two bounds. This means that the same set of trees used to calculate the upper bound can also be utilized to determine the lower bound. Instead of maintaining two separate sets of scenario trees for the upper and lower bounds separately, employing the "common random number" technique can effectively reduce variance and enhance the efficiency in estimation of the optimality gap in MSSP. This technique involves constructing n_g independent scenario trees $\Gamma(\vec{\xi}^{1,i})$. For each scenario tree, we derive a point estimator W_i for the upper bound using Equation (3.15) and the methods outlined in Section 3.3.2.1, along with a lower bound ℓ_i as described in Section 3.3.2.2. The difference

$G_i = W_i - \ell_i$ serves as an estimator for the optimality gap. Furthermore, let $\overline{G}$ and s_g^2 be the sample mean and sample variance of G_i. Since $\Gamma(\vec{\xi}^{1,i})$ are constructed on an i.i.d. basis, we can establish an approximate $(1-\alpha)$-level confidence interval for the optimality gap using the formula:

$$\left[0, \overline{G} + \frac{t_{n_g-1,\alpha} s_g}{\sqrt{n_g}}\right], \tag{3.18}$$

where $t_{n_g-1,\alpha}$ represents the $1-\alpha$ quantile of the t-distribution with $n_g - 1$ degrees of freedom.

In Algorithm 3 we summarize the procedure for finding the optimality gap using the MRP estimators for MSSP with common random number streaming.

Algorithm 3 MRP Confidence Interval Estimation

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

3.4. Sequential Sampling Procedure

Bayraksan, Morton and Pierre-Louis [7, 8] introduced sequential sampling algorithms that develop stopping rules for Monte Carlo sampling-based algorithms. These rules ensure high-quality solutions with high probability once met. Unlike the MRP or A2RP estimators, which compute the optimality gap for a fixed solution, the sequential sampling algorithms evaluate the optimality gap in two-stage stochastic programming across a sequence of feasible solutions converging towards an optimal limit point. This sequence can be generated by progressively solving sampling problems with increasing sample sizes or by employing any other effective method. The sequential sampling algorithms assess the optimality gap for each solution in the sequence. If this gap falls below a predefined threshold, the process terminates with the current candidate solution. Otherwise, the evaluation continues with the next solution, making use of a larger sample size.

The Sequential Sampling Procedure for two-stage stochastic programming is as follows, with input parameters $h > h' > 0, \varepsilon > \varepsilon' > 0$, and $p > 0$. Here, p is used to control the sample size. Given K, it is possible to find p that minimizes n_K. However, as shown in [7], given p, the variation in sample size is actually moderate for different K. The parameter h is used to control the desired width of the confidence interval, while 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 larger sample size. In addition, ϵ, ϵ' is used here to ensure finite termination and theoretical convergence, and we require $h - h'$ to some extent to be proportional to $\epsilon - \epsilon'$. The candidate solution is given as $\{\widehat{x}_k\}$ with at least one limit point. The parameter k_f controls the resample frequency.

Sequential Sampling Procedure for Two-Stage SP:

0. (Initialization) Set $k = 1$, compute the sample size n_k for the current iteration with

$$n_k = \left\lceil \frac{1}{(h-h')^2}(c_p + 2p\ln^2 k) \right\rceil, \qquad c_p = \max\left\{ 2\ln\left(\sum_j j^{-p\ln j}/\sqrt{2\pi\alpha} \right), 1 \right\},$$

   and sample observations $\xi^1, \xi^2, \ldots, \xi^{n_k}$.

(1) Compute the point estimator G_k and the sample variance estimator s_k^2 for the optimality gap using (3.11) and (3.13).

(2) If $\{G_k \leqslant h's_k + \varepsilon'\}$, then terminate and output the candidate solution $\widehat{x}_k$ and a one-sided CI on the optimality gap,

$$[0, hs_T + \varepsilon].$$

(3) Otherwise, set $k = k + 1$ and re-calculate n_k. If k_f divides k, sample new observations $\xi^1, \xi^2, \ldots, \xi^{n_k}$ that are independent of the previous iterations. Otherwise, sample additional $n_k - n_{k-1}$ observations $\xi^{n_{k-1}+1}, \xi^{n_{k-1}+2}, \ldots, \xi^{n_k}$ from the distribution of ξ. Return to the first step.

Since sequential sampling procedures involve estimating optimality gaps in Step 1, the work in [7, 8] is not directly applicable to multistage situations. In the following subsections, we outline a straightforward extension to adapt these procedures for multistage contexts. The proposed adaptation includes the simplification of the Multistage Stochastic Programming (MSSP) problem by the relaxation of non-anticipativity constraints, thus regarding the MSSPs as two-stage problems.

While the relaxation simplifies the application of sequential sampling methods, it is important to note that due to these relaxed constraints, the confidence intervals obtained are in general not as tight as those derived with the original, more complex multistage models.

3.4.1. Relative Width Sequential Sampling. To adapt the relative width sequential sampling procedure for a multi-stage setting, it is necessary to develop methods for constructing point estimators and sample variance estimators that are appropriate for multi-stage scenario trees, with a set of sample scenarios. Point estimators can be formulated by determining an upper bound for the function value following the approach used in MRP estimators, together with a lower bound that corresponds to the optimal value of the MSSP problem for the given sampled scenarios.

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 $\{\boldsymbol{\xi}_t\}_{t=2}^{T}$ 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 (3.2), except at the root node. In essence, the discrete MSSP problem(3.7) for the horsetail scenario tree is equivalent to the following problem,

$$\hat{z}^* = \min_{x_1 \in X_1} \left\{ \frac{1}{n_k} \sum_i \min_{x_2,\ldots,x_T} \psi'_T(\vec{x}^T; \vec{\xi}^{T,i}) \right\}, \tag{3.19}$$

where

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

Since this is a relaxation of the original problem (3.2), we can expect the optimal solution for (3.19) to be a valid (but possibly not consistent) statistical lower bound for problem (3.2). Algorithm 4 includes the procedure for finding the point estimator and the sample variance estimator for a horsetail scenario tree.

Once we establish a method to compute point estimators and sample variance estimators for the MSSP, these estimators can be integrated into the sequential sampling procedure to derive stopping criteria for evaluating a series of candidate solutions within the MSSP framework. Algorithm 5 outlines our implementation of a multi-stage adaptation of the sequential sampling procedure in [7]

Algorithm 4 Optimality Gap Estimation with Horsetail Scenario Tree

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

Algorithm 5 Sequential Sampling

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.
for $k = 1$ **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 $\vec{\xi}^{T,1}, \vec{\xi}^{T,2}, \ldots, \vec{\xi}^{T,n_k}$
    else
        Independently sample $n_k - n_{k-1}$ new scenarios $\vec{\xi}^{T,n_{k-1}+1}, \ldots, \vec{\xi}^{T,n_k}$ and bundle them with the previously sampled scenarios $\vec{\xi}^{T,1}, \vec{\xi}^{T,2}, \ldots, \vec{\xi}^{T,n_{k-1}}$
    Use Algorithm 4 to compute the optimality gap estimator G_k and the variance estimator s_k for $\widehat{x}_{1,k}$ using the sampled scenarios
    if $G_k \leqslant h' s_k + \epsilon'$ **then**
        Set $K = k$; **break**
return $\widehat{x}_{1,K}$, $(1-\alpha)$-level CI $[0, hs_K + \epsilon]$

The main theorem in [7] states that for input parameters h, ϵ and ϵ', the returned results for a two-stage problem satisfies

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

In the context of multi-stage problems, while the underlying inequality remains consistent, the resulting confidence interval is typically broadened due to the relaxation of the non-anticipativity

constraint. This is because the estimators derived from horsetail scenario trees tend to be conservative by nature. Further research is needed to develop methods for constructing sample scenario trees that take into account the non-anticipativity constraints, with the aim of improving the accuracy of both point estimators and sample variance estimators for the optimality gap.

3.4.2. Fixed Width Sequential Sampling. For relative width sequential sampling method as described in Algorithm 5, the stopping criteria are met when the optimality gap of the candidate solution 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 [8], who later proposed an alternative stopping rule with the aim of finding 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 decreases to zero as $n \to 0$, their proposed stopping rule returns the candidate solution when $G_n(x)+\nu_n(x)+h(n) \leqslant \epsilon$. For an MSSP adaptation, the point estimator $G_n(x)$ can be computed in the same way as in Algorithm 5, 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 of a t-distribution with $n-1$ degrees of freedom. The procedure for fixed-width sequential sampling with horsetail scenario trees is in Algorithm 6.

Algorithm 6 Fixed Width Sequential Sampling

Input: Sample size schedules $\{m_k\}$, $\{n_k\}$, inflation factor $h(n)$, resampling frequency k_f^n, threshold ϵ.
Output: Candidate first stage solution $\widehat{x}_{1,K}$ whose estimated optimality gap falls below ϵ.
for $k = 1$ **to** MaxIter **do**
    Independently sample a scenario tree with m_k leaf nodes and solve the approximate problem 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 sample $n_k - n_{k-1}$ new scenarios $\vec{\xi}^{T,n_{k-1}+1}, \ldots, \vec{\xi}^{T,n_k}$ and bundle them with the previously sampled scenarios $\vec{\xi}^{T,1}, \vec{\xi}^{T,2}, \ldots, \vec{\xi}^{T,n_{k-1}}$
    Use Algorithm 4 to compute the optimality gap estimator G_k and the variance estimator s_k for $\widehat{x}_{1,k}$ using the sampled scenarios
    if $G_k + \frac{t_{n-1,\alpha} s_n(x)}{\sqrt{n}} + h(n) \leqslant \epsilon$ **then**
        Set $K = k$; **break**
return $\widehat{x}_{1,K}$

3.4.3. Software Particulars. A version of the Multiple Replication Procedures and the Sequential Sampling Procedures for estimating the optimality gap for two-stage and multi-stage stochastic programming problems is available within the MPI-SPPY software. 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/. Both MRP 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 MRP 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 [13, 39] 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 MRP program 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 MRP program, so the upper bound software used by any method supported by mpi-sppy can create the solution. The mpi-sppy library has heuristics for finding computing upper bounds given a scenario tree and does not require relatively complete recourse. The sequential sampling software does not require $\widehat{x}$ as input because it is one of the outputs.

CHAPTER 4

Uncertainty Quantification in Optimization: Bootstrap and Bagging Methods

4.1. Introduction

Many research articles on stochastic programming begin with the assumption that uncertain data come from a known distribution F, and that one is able to sample as many samples as they want from the distribution. In this chapter, we describe methods that do not rely on this assumption, but use only sampled data to obtain both a consistent sample-average solution and a consistent estimate of confidence intervals for the optimality gap. The underlying distribution whence the samples come is not required.

For consistent estimators of a confidence interval around the objective function value, we rely on bootstrap and bagging estimators. In addition, we consider various combinations of distribution estimation and resampling (bootstrap and bagging) for obtaining samples used to estimate the optimality gap and a confidence interval (CI) for it. We make three contributions to ongoing research in data-driven stochastic programming: a) most of the combinations of distribution estimation and resampling result in algorithms that have not been published before, b) within the algorithms, we describe innovations that improve performance, and c) we provide open-source software implementations of the algorithms. The end result is a significant improvement in the ability to use data to estimate a confidence interval around the objective function value for a candidate solution to an optimization problem under uncertainty.

For this chapter, we mainly focus on the abstract form of a stochastic optimization problem using

$$\min_x E_\xi[g(x,\xi)] \tag{4.1}$$

We will consider examples with explicit constraints, but for now, we use notation where they are implicit. The decision vector is x and the vector of uncertain data is ξ. Data known with certainty,

along with constraints, are captured in the specification of the function g. Our interest is when there is a sample $\mathcal{Z}_N = \{\mathfrak{z}_i,\ i = 1, \ldots, N\}$ that can be used for a given $x = \widehat{x}$ to estimate the optimality gap and its confidence interval. The subscript i indicates the i^{th} vector; we never reference vector elements in this chapter.

The optimality gap $\mathcal{G}_{\widehat{x}}$ for a decision $\widehat{x}$ with respect to a distribution $\mathbb{Q}$ is expressed as:

$$\mathcal{G}_{\widehat{x}}(\mathbb{Q}) = E_{\xi\sim\mathbb{Q}}[g(\widehat{x}, \xi)] - \min_x E_{\xi\sim\mathbb{Q}}[g(x, \xi)].$$

For ease of notation, when $\mathbb{Q}$ is the empirical distribution derived from some sample set $S_N = \{s_i\}_{i=1}^N$, we allow $\mathcal{G}_{\widehat{x}}$ to directly accept S_N as input, so that

$$\mathcal{G}_{\widehat{x}}(S_N) = \frac{1}{N}\sum_{i=1}^{N} g(\widehat{x}, s_i) - \min_x \frac{1}{N}\sum_{i=1}^{N} g(x, s_i).$$

We drop the subscript $\widehat{x}$ when it does not cause confusion, and write the optimality gap function as $\mathcal{G}(\cdot)$.

4.1.1. Literature. Bootstrap has been widely used for statistical inference since it was first proposed by [29]. It can be used to construct confidence intervals when the underlying true distribution remains unknown; see [30], [91], and [22] for a comprehensive introduction.

In the area of stochastic programming, an early work in [41] used bootstrap to develop stopping rules for the Stochastic Decomposition algorithm. Subsequent studies, such as those of [31] and [59], proposed the use of bootstrap and related resampling methods to derive confidence intervals for the optimal function value. Anitescu et al. [2] discussed some theoretical properties of bootstrap confidence intervals for stochastic programming. In contrast to the primary goal of a confidence interval on the optimality gap, various software packages compute confidence intervals for the value of a particular solution, which is an upper bound on the optimal solution; e.g., the `sddp.jl` software [26]. Another use of bootstrap in stochastic programming is in Parmest [55], which is a tool for model-based estimation of unobservable parameter values from experiments by minimizing the squared error of model prediction. (In the use cases envisioned for the tool, the models involve chemical processes, but it could be any model.) The tool also provides three methods for estimating confidence intervals around the estimated parameter vector, one of which uses classical bootstrap.

The tool can be used to create scenarios for optimization under uncertainty and could provide inputs for the estimation of confidence intervals for the optimality gap.

The classical bootstrap estimates a statistic of interest, say $\alpha(F)$, by its empirical version $\alpha(F_n)$. There is a literature that discusses the properties of smoothed bootstrap, where the discrete distribution F_n is replaced with a smoothed distribution for estimation; see, for example, [23, 96] for theoretical discussions, and [35, 63] for application examples. The application of smoothed bootstrap in the area of optimization remains limited.

The bagging method, also known as bootstrap aggregation, was proposed in [11] and has been widely used in the machine learning community to produce accurate and robust predictions by aggregating the predictions of multiple models, each using bootstrap samples from the original dataset; see [12] for theoretical analysis. Smoothed bagging for classification and regression problems was proposed in [83], which included added noise in the resampled data. More recently, [59] proposed the use of the bagging method to construct confidence intervals for a candidate solution in stochastic programming.

Despite the limited application of smoothed bootstrap in optimization, there are works on combining probability density estimation with stochastic optimization problems. For example, [48] used the KM estimator to construct an empirical cumulative distribution function for censored data in the newsvendor problem, and [76] used Markov chain Monte Carlo methods with kernel density estimation algorithms to build a non-parametric importance sampling distribution for the recourse function.

4.2. Bootstrap and Bagging Method

Below we discuss the bootstrap and bagging procedures for estimating confidence intervals on optimality gaps for general stochastic programming problems. We do not describe algorithms for finding a candidate solution $\widehat{x}$ because that is already the subject of a large literature; see, e.g., [75].

Given a candidate solution $\widehat{x}$, we are interested in deriving a confidence interval for the optimality gap. One may also be interested in deriving a confidence interval for the optimal function value z^*, or for the function value for the candidate solution $E_\xi[g(\widehat{x}, \xi)]$. It turns out that all these estimators can be derived in a similar way as described in the following subsections.

For ease of notation, we assume that we can solve $\min_x \frac{1}{N}\sum_{i=1}^{N} g(x, d_i)$ exactly by simply passing the problem to a solver.

4.2.1. Classical Bootstrap Method. The bootstrap method is used for estimating the optimality gap in [31]. In the original paper, the bootstrap method is applied to find a confidence interval for the optimal function value z^*, but the same method can be used for the optimality gap with a minor modification.

At each bootstrap iteration, we resample from set $\mathcal{Z}_N$ to obtain a bootstrap sample $\widetilde{\mathcal{Z}}_N^b$, and compute the corresponding optimality gap under the set $\widetilde{\mathcal{Z}}_N^b$,

$$\mathcal{G}_{\widehat{x}}(\mathcal{Z}_N) = \frac{1}{N}\sum_{i=1}^{N} g(\widehat{x}, \mathfrak{z}_i) - \min_x \frac{1}{N}\sum_{i=1}^{N} g(x, \mathfrak{z}_i)$$

Let Z_N and $\mathcal{Z}_N$ represent the random set and its realization, respectively. Each $\mathfrak{z}_i$ in $\mathcal{Z}_N$ is a realization from the distribution F. And let $\widetilde{Z}_N^b$ be the random set whose elements obey the empirical distribution of the set $\mathcal{Z}_N$. The classical bootstrap method is based on the theoretical validation of the asymptotic similarity ([91]) between the two distributions, $\mathcal{G}(Z_N) - \mathcal{G}(F)$ and $\mathcal{G}(\widetilde{Z}_N^b) - \mathcal{G}(\mathcal{Z}_N)$, with the latter conditioned on one realization of the random variable Z_N, which is the random sample $\mathcal{Z}_N$. See [31, Corollary 4.1, Corollary 5.1] for a proof.

Algorithm 7 describes the procedure for finding an approximate confidence interval using the classical bootstrap procedure, where the CI is centered on $\mathcal{G}(\mathcal{Z}_N)$, the gap associated with the set $\mathcal{Z}_N$, and the quantile of bootstrap sampled gaps is used to derive the limits.

Algorithm 7 Classical Bootstrap

1: **Input:** A sample $\mathcal{Z}_N = \{z_i\}_{i=1}^N$, number of batches B, and a candidate solution $\widehat{x}$
2: Compute the optimality gap associated with the set $\mathcal{Z}_N$

$$\mathcal{G}_{\widehat{x}}(\mathcal{Z}_N) = \frac{1}{N}\sum_{i=1}^{N} g(\widehat{x}, z_i) - \min_x \frac{1}{N}\sum_{i=1}^{N} g(x, z_i)$$

3: **for** $b \leftarrow 1$ **to** B **do**
4: Resample from $\mathcal{Z}_N$ to get the bootstrap set $\widetilde{\mathcal{Z}}_N^b = \{\tilde{z}_1^b, \ldots, \tilde{z}_N^b\}$
5: Compute the associated gap

$$\mathcal{G}(\widetilde{\mathcal{Z}}_N^b) = \frac{1}{N}\sum_{i=1}^{N} g(\widehat{x}, \tilde{z}_i^b) - \min_x \frac{1}{N}\sum_{i=1}^{N} g(x, \tilde{z}_i^b)$$

6: Compute the upper $1-\alpha$-quantile $\varrho_{1-\alpha}$ and lower α-quantile ϱ_α for $\{\mathcal{G}(\widetilde{\mathcal{Z}}_N^b) - \mathcal{G}(\mathcal{Z}_N)\}$
7: **return** $[\mathcal{G}(\mathcal{Z}_N) - \varrho_{1-\alpha}, \mathcal{G}(\mathcal{Z}_N) - \varrho_\alpha]$ as the $(1-2\alpha)$ confidence interval for the optimality gap $\mathcal{G}(F)$

If we replace the optimality gap $\mathcal{G}_{\widehat{x}}(\mathcal{Z}_N)$ with the optimal function value under the set $\mathcal{Z}_N$, that is, $\min_x \frac{1}{N}\sum_{i=1}^N g(x, \mathfrak{z}_i)$, and use $\min_x \frac{1}{N}\sum_{i=1}^N g(x, \widetilde{\mathfrak{z}}_i^b)$ in place of $\mathcal{G}(\widetilde{\mathcal{Z}}_N^b)$, then the above algorithm can be used to find an approximate confidence interval for the optimal function value z^*. The same argument applies to all the algorithms in the following subsections.

Note that there are a few variations on the classical bootstrap method, in that different metrics can be used to derive the confidence interval from the resampled gaps $\mathcal{G}(\widetilde{\mathcal{Z}}_N^b)$. For example, instead of using the quantiles of $\mathcal{G}(\widetilde{\mathcal{Z}}_N^b)$ in one way or another, one can also use $\mathcal{G}(\widetilde{\mathcal{Z}}_N^b)$ to fit a standard normal confidence interval. As in [29], one can compute the variance of $\mathcal{G}(\widetilde{\mathcal{Z}}_N^b)$, denoted as s^2, and return $[\mathcal{G}(\mathcal{Z}_N) - z_{1-\alpha}s, \mathcal{G}(\mathcal{Z}_N) + z_{1-\alpha}s]$ as the CI, with $z_{1-\alpha}$ being the quantile of the standard Gaussian variable.

4.2.2. Extended Bootstrap Method. Theoretically, the consistency of the estimated confidence interval in Algorithm 7 may no longer stand if the stochastic program does not have a unique solution. An extended two-stage bootstrap method that tackles such situations is developed in [31] and shown in Algorithm 8. Note that the extended bootstrap method is only applicable if we are able to sample from the unknown distribution F, which may not be possible in various settings, but we include the algorithm here for completeness. The consistency of the extended bootstrap method was proven in [31, Lemma 5.2] for the CI estimator for the optimal function value. As with the classical bootstrap method, the same algorithm can be used to find a confidence interval for the optimal function value by replacing the optimality gap function $\mathcal{G}(\cdot)$ with the corresponding empirical optimal function value.

Algorithm 8 Extended Bootstrap

1: **Input:** number of batches B, significance level α, and a candidate solution $\widehat{x}$
2: **for** $b \leftarrow 1$ **to** B **do**
3: Independently sample from F to get set $\mathcal{Z}_b = \{z_1, \ldots, \widehat{z}_N\}$ and compute the corresponding optimality $\mathcal{G}(\mathcal{Z}_b)$
4: Resample from $\mathcal{Z}_b$ to get $\widetilde{\mathcal{Z}}_b = \{\widetilde{z}_1, \ldots, \widetilde{z}_N\}$ and compute $\mathcal{G}(\widetilde{\mathcal{Z}}_b)$
5: Compute the upper $1-\alpha/2$-quantile $\varrho_{1-\alpha/2}$ and lower $\alpha/2$-quantile $\varrho_{\alpha/2}$ for $\{\mathcal{G}(\widetilde{\mathcal{Z}}_b) - \mathcal{G}(\mathcal{Z}_b)\}$
6: Independently sample from F to get $\overline{\mathcal{Z}}_N$ and $\underline{\mathcal{Z}}_N$, each of size N
7: Compute the center of the confidence interval $G = 2 * \mathcal{G}(\overline{\mathcal{Z}}_N \cup \underline{\mathcal{Z}}_N) - \mathcal{G}(\overline{\mathcal{Z}}_N)$
8: **return** $[G - \varrho_{1-\alpha/2}, G - \varrho_{\alpha/2}]$ as the $(1-\alpha)$ confidence interval for the optimality gap

4.2.3. Subsampling. The subsampling method for estimating the confidence interval of z^* was briefly mentioned in [31]. The procedure is essentially the same as the classical bootstrap

method, except that a smaller subsampling size is adopted for resampling. Algorithm 9 details the procedure for finding the confidence interval of the optimality gap, and the consistency of the estimator is shown in [31, Section 5.3].

Algorithm 9 Subsampling

1: **Input:** A sample $\mathcal{Z}_N = \{z_i\}_{i=1}^N$, number of batches B, sub-sample size k, significance level α, and a candidate solution $\widehat{x}$
2: Compute the optimality gap associated with the set $\mathcal{Z}_N$

$$\mathcal{G}_{\widehat{x}}(\mathcal{Z}_N) = \frac{1}{N}\sum_{i=1}^{N} g(\widehat{x}, z_i) - \min_x \frac{1}{N}\sum_{i=1}^{N} g(x, z_i)$$

3: **for** $b \leftarrow 1$ **to** B **do**
4: Resample from $\mathcal{Z}_N$ without replacement to get set $\widetilde{\mathcal{Z}}_b$ of size k, $\widetilde{\mathcal{Z}}_b = \{\widetilde{z}_1, \ldots, \widetilde{z}_k\}$
5: Compute $\mathcal{G}(\widetilde{\mathcal{Z}}_b)$
6: Compute the upper $1-\alpha/2$-quantile $\varrho_{1-\alpha/2}$ and lower $\alpha/2$-quantile $\varrho_{\alpha/2}$ for $\{\sqrt{k}(\mathcal{G}(\widetilde{\mathcal{Z}}_b) - \mathcal{G}(\mathcal{Z}_N))\}$
7: **return** $[\mathcal{G}(\mathcal{Z}_N) - \sqrt{\frac{1}{N}}\varrho_{1-\alpha/2}, \mathcal{G}(\mathcal{Z}_N) - \sqrt{\frac{1}{N}}\varrho_{\alpha/2}]$ as the $(1-\alpha)$ CI for the optimality gap

4.2.4. Bagging. A bagging procedure proposed in [59] approximates the confidence interval of the optimality gap without the need to sample for the entire population and without the need for a unique solution. Compared with the bootstrap method, the bagging method is known to reduce variance in general applications.

We note here that the bagging procedure described in [59] bears some similarity to the subsampling method in Algorithm 9, as in both cases a smaller subsampling size is used. The difference lies in how the confidence interval is formed: The subsampling method directly uses the quantiles of the subsamples to derive a confidence interval, whereas the bagging procedure uses an empirical version of the infinitesimal jackknife estimator for variance estimation.

Algorithm 10 illustrates the procedure for using bagging to find an approximate confidence interval for the optimality gap. The original algorithm in [59] focuses on finding CI for the optimal function value, and Algorithm 10 is a small variation of the original one.

4.3. Asymptotic Theory

In this section, we discuss the asymptotic convergence of the bootstrap method for a set of convex optimization problems where the constraints can be expressed as the expectation over

Algorithm 10 Bagging-based Sampling

1: **Input:** A sample $\mathcal{Z}_N$, number of bags B, bag sample size k, significance level α, and a candidate solution $\widehat{x}$
2: **for** $b \leftarrow 1$ **to** B **do**
3: Resample from $\mathcal{Z}_N$ to get a bagging set of size k, $\widetilde{\mathcal{Z}}_N^b = \{\tilde{z}_1^b, \ldots, \tilde{z}_k^b\}$
4: Compute $\mathcal{G}(\widetilde{\mathcal{Z}}_N^b) = \frac{1}{k}\sum_{i=1}^k g(\widehat{x}, \tilde{z}_i^b) - \min_x \frac{1}{k}\sum_{i=1}^k g(x, \tilde{z}_i^b)$
5: Compute the mean of $\mathcal{G}(\widetilde{\mathcal{Z}}_N^b)$ as the center of the confidence interval:

$$G = \frac{1}{B}\sum_{b=1}^{B} \mathcal{G}(\widetilde{\mathcal{Z}}_N^b)$$

6: Compute the error term:

$$\widetilde{\sigma}^2 = \begin{cases} \sum_{i=1}^n \widehat{\mathrm{cov}}_i^2 & \text{if with replacement} \\ \frac{n^2}{(n-k)^2}\sum_{i=1}^n \widehat{\mathrm{cov}}_i^2 & \text{if without replacement} \end{cases},$$

where

$$\widehat{\mathrm{cov}}_i = \frac{1}{B}\sum_{b=1}^{B}(N_i^b - k/n)(\mathcal{G}_{\mathrm{gap}}(\widetilde{\mathcal{Z}}_N^b) - G),$$

and N_i^b is the number of times the i^{th} element of $\mathcal{Z}_N$ appears in $\widetilde{\mathcal{Z}}_N^b$.
7: Return $[G - z_{1-\alpha/2}\widetilde{\sigma}, G + z_{1-\alpha/2}\widetilde{\sigma}]$ as the $(1-\alpha)$ CI for the optimality gap $\mathcal{G}(F)$, with $z_{1-\alpha/2}$ being the $(1-\alpha/2)$ quantile of a standard normal variable.

certain functions as in [92]:

$$\begin{aligned} \min_{x\in S} \quad & E_\xi[g_0(x,\xi)] \\ s.t. \quad & E_\xi[g_i(x,\xi)] \leqslant 0, \qquad i = 1,\ldots,r \end{aligned} \tag{4.2}$$

Many stochastic problems can be represented with this formulation (see [32, 61]). For general results on the asymptotic convergence of bootstrap methods applicable to a wider range of stochastic programming challenges, we refer the reader to [31].

4.3.1. Notations. We begin by establishing some notation that will facilitate the subsequent analysis. Specifically, we use F to denote the unknown distribution of the random variable ξ, and $\widehat{F}_N$ to denote the empirical distribution of a set of observations $\mathcal{Z}_N$. We let x^* be the optimal solution for the SP problem and let z^* be the optimal function value. In addition, let $\widehat{x}_N$ be the optimal value for the approximated SP problem with set $\mathcal{Z}_N$, so that $\widehat{x}_N = \min_x \frac{1}{N}\sum_j g_0(x, \mathfrak{z}_j)$, and let $\widehat{z}_N$ be the corresponding function value $\widehat{z}_N = \frac{1}{N}\sum_j g_0(\widehat{x}_N, \mathfrak{z}_j)$.

For simplicity we define

$$f_i(x) = E_\xi[g_i(x,\xi)], \qquad i = 0,\dots,r.$$

Then problems we are considering can be rewritten as

$$\begin{aligned} \min_{x\in S} \quad & f_0(x) \\ s.t. \quad & f_i(x) \leqslant 0, \qquad i = 1,\dots,r \end{aligned} \tag{4.3}$$

When given a set of observations $\mathcal{Z}_N = \{\mathfrak{z}_i,\ i = 1,\dots,N\}$, the original problem(4.2) can be approximated using a sample average approximation (SAA),

$$\begin{aligned} \min_{x\in S} \quad & E_{\widehat{F}_N(\xi)}[g_0(x,\xi)] = \frac{1}{N}\sum_{j=1}^{N} g_0(x,\mathfrak{z}_j) \\ s.t. \quad & E_{\widehat{F}_N(\xi)}[g_i(x,\xi)] \leqslant 0, \qquad i = 1,\dots,r. \end{aligned} \tag{4.4}$$

We introduce the notation

$$\psi_{i,N}(x) = E_{\widehat{F}_N(\xi)}[g_i(x,\xi)] \qquad i = 0,\dots,r$$

and express the approximated problem as

$$\begin{aligned} \min_{x\in S} \quad & \psi_{0,N}(x) \\ s.t. \quad & \psi_{i,N}(x) \leqslant 0, \qquad i = 1,\dots,r. \end{aligned} \tag{4.5}$$

here $\psi_{i,N}(x)$ implicitly depends on the set $\mathcal{Z}_N$.

In addition, given a bootstrap sample $\widetilde{\mathcal{Z}}_b = \{\widetilde{\mathfrak{z}}_1,\dots,\widetilde{\mathfrak{z}}_N\}$ that is i.i.d. sampled from the empirical distribution $\widehat{F}_N$, the corresponding optimization problem is

$$\begin{aligned} \min_{x\in S} \quad & \frac{1}{N}\sum_{j=1}^{N} g_0(x,\widetilde{\mathfrak{z}}_j) \\ s.t. \quad & \frac{1}{N}\sum_{j=1}^{N} g_i(x,\widetilde{\mathfrak{z}}_j) \leqslant 0, \qquad i = 1,\dots,r \end{aligned} \tag{4.6}$$

We denote

$$\widetilde{\psi}_{i,N}(x) = \frac{1}{N}\sum_{j=1}^{N} g_i(x, \widetilde{\mathfrak{z}}_j), \qquad i = 0, \ldots, r$$

to rewrite the bootstrap optimization problem as

$$\begin{aligned} \min_{x \in S} \quad & \widetilde{\psi}_{0,N}(x) \\ s.t. \quad & \widetilde{\psi}_{i,N}(x) \leqslant 0, \qquad i = 1, \ldots, r. \end{aligned} \tag{4.7}$$

The following notations are utilized throughout the theoretical framework: $C_m(S)$ represents the space of continuous functions defined on S with values in $\mathbb{R}^m$. For brevity, when $m = 1$, we denote this space simply as $C(S)$. The Cartesian product space $\mathcal{L}$ is defined as $C(S) \times \cdots \times C(S)$, which constitutes the product of $r + 1$ such spaces. The subset $\mathcal{H}$ within $\mathcal{L}$ comprises vectors $\xi = (\xi_0, \ldots, \xi_r)$ whose components are convex functions in an open neighborhood of S. Lastly, the Lagrangian function is given by $L(x, \lambda, \zeta) = \xi_0(x) + \sum_{i=1}^{r} \lambda_i \zeta_i(x)$, where x represents the decision variables, λ the vector of Lagrange multipliers, and ζ the vector of functions from $\mathcal{H}$.

4.3.2. Main Theorem. The main theorem is built upon the following assumptions::

a. S is compact and convex.

b. $g_0, \ldots, g_r$ are real valued and convex. The convexity of g_i implies the convexity of f_i and $\psi_{i,N}$, and, in turn, implies that μ and Y_n are in space $\mathcal{L}$.

c. The Slater condition holds for program (4.3), which implies strong duality for convex optimization.

d. The program (4.3) has a unique optimal solution x^* and a unique Lagrange multiplier vector λ^*

e. For every $x \in S$ the function $g_i(x, \cdot)$ is measurable for each i

f. For each i, for some point $x_0 \in S$, the expectation $E_\xi |g(x_0, \xi)|^2$ is finite

g. For each function g_i, there exists a function b_i, such that $E_\xi[b_i(\xi)^2]$ is finite and for all $x, y \in S$,

$$|g_i(x, \xi) - g_i(y, \xi)| \leqslant b_i(\xi)\|x - y\|$$

In addition, we need an assumption that corresponds to assumption (d) for the bootstrap samples.

h. The approximated program (4.4) has a unique optimal solution $\widehat{x}_N$ and a unique Lagrange multiplier vector $\widehat{\lambda}$ with probability one.

The main theorem states as follows:

THEOREM 4.3.1. *Suppose that Assumptions (a)-(h) hold. Then*

$$\liminf_{N,n_B\to\infty} \Pr\left(\mathcal{G}(F) \in [\varrho_\alpha, \varrho_{1-\alpha}]\right) \geqslant 1 - 2\alpha. \tag{4.8}$$

Here ϱ_α and $\varrho_{1-\alpha}$ are the output from Algorithm 7.

We will show the proof of this theorem at the end of the next section.

4.3.3. Preliminaries for the Proof of Theorem 4.3.1. Before proving the final result, we need to introduce several additional notations and intermediate results that are required for the proof.

We follow the work of [92] and work directly with the functions of programs (4.3) and (4.5) instead of the corresponding empirical distributions. That is, define $\mu = (f_0, f_1, \ldots, f_r)$ to be the Cartesian product of functions f_i. Under assumption (g), μ is in $\mathcal{L}$, the Cartesian product of the space of continuous functions, and thus it belongs to the space $C_{r+1}(S)$. Similarly, define $Y_N = (\psi_{0,N}, \ldots, \psi_{r,N}) \in \mathcal{L}$.

The key idea in the proof is to treat stochastic programs as elements of the function space $\mathcal{L}$, and to use an extended delta method on a functional that maps a program to the corresponding optimality gap.

To that end, we introduce a linear functional $\bar{\varphi} : \mathcal{L} \to \mathbb{R}$ that maps a vector $\zeta = (\zeta_0, \zeta_1, \ldots, \zeta_r)$ to the corresponding function value for the candidate solution $\widehat{x}$,

$$\bar{\varphi}(\zeta) = \zeta_0(\widehat{x})$$

Similarly $\bar{\phi}(\cdot) : \mathcal{L} \to \mathbb{R}$ is defined as a functional that returns the optimal function value of the corresponding stochastic program (4.3), so

$$z* = \bar{\phi}(\mu), \qquad \widehat{z}_N = \bar{\phi}(Y_N)$$

We use the notation γ to present a function that maps a vector in space $\mathcal{L}$ to the corresponding optimality gap. That is, given a vector $\zeta = (\zeta_0, \zeta_1, \ldots, \zeta_r) \in \mathcal{L}$,

$$\gamma(\zeta) = \bar{\varphi}(\zeta) - \bar{\phi}(\zeta). \tag{4.9}$$

In this way, the actual optimality gap for a candidate solution $\widehat{x}$ can be expressed as $\mathcal{G}(F) = \gamma(\mu)$, whereas the optimality gap under the set $\mathcal{Z}_N$ is $\mathcal{G}(\mathcal{Z}_N) = \gamma(Y_N)$.

THEOREM 4.3.2 (Converge in sampling). *Suppose Assumptions (a)-(g) hold. Then*

$$n^{1/2}(\gamma(Y_n)) - \gamma(\mu)) \xrightarrow{D} \mathcal{N}(0, \varsigma^2). \tag{4.10}$$

Here $\mathcal{N}(0, \varsigma^2)$ is a normal variable with mean 0 and variance

$$\varsigma^2 = E_\xi[K(\widehat{x}, x^*, \lambda^*, g, \xi)]^2 - \{E[K_\xi(\widehat{x}, x^*, \lambda^*, g, \xi)]\}^2, \tag{4.11}$$

with $K(\widehat{x}, x^*, \lambda^*, g, \xi)$ being a random variable with respect to ξ,

$$K(\widehat{x}, x^*, \lambda^*, g, \xi) = g_0(\widehat{x}, \xi) - g_0(x^*, \xi) - \sum_i \lambda_i^* g_i(x^*, \xi).$$

PROOF. We start by showing that $\gamma(\zeta)$ is Hadamard directional differentiable at μ. $\bar{\phi}(\zeta)$ is Hadamard directional differentiable by (4.19) in Theorem 4.3.5. Since $\bar{\varphi}(\zeta)$ is a linear functional, it is easy to see that $\bar{\varphi}(\zeta)$ is Hadamard differentiable at μ with

$$\bar{\varphi}'_\mu(\zeta) = \bar{\varphi}(\zeta). \tag{4.12}$$

Hence, by the definition of $\gamma(\zeta)$, it is also Hadamard differentiable at μ tangentially to the set $\mathcal{H}$. And for all the vectors ζ in the Bouligand Tangent cone $T_\mu(\mathcal{H})$,

$$\gamma'_{\mu,\mathcal{H}}(\zeta) = \bar{\varphi}(\zeta) - L(x^*, \lambda^*, \zeta) \tag{4.13}$$

Now under assumption (a)-(b) and (e)-(g), by Thm 4.3.8, we have $n^{1/2}(Y_N - \mu)$ converging to a random element W in $\mathcal{L}$, and W follows a vector-valued Gaussian process. That is, for any finite collection $x_1, \ldots, x_m$ in S, $\{W(x_1), \ldots, W(x_m)\}$ forms a jointly Gaussian. In particular, we have

$W(x^*) = (w_0(x^*), \ldots, w_r(x^*))$ and $W(\widehat{x}) = (w_0(\widehat{x}), \ldots, w_r(\widehat{x}))$ are jointly Gaussian. Therefore using delta method as in [92] (Thm 2.1), we have

$$n^{1/2}(\gamma(Y_N) - \gamma(\mu)) \xrightarrow{D} \gamma'_{\mu,\mathcal{H}}(W),$$

By (4.13), this implies that

$$n^{1/2}(\gamma(Y_N) - \gamma(\mu)) \xrightarrow{D} \bar{\varphi}(W) - L(x^*, \lambda^*, W),$$

Equation (4.10) then follows by recognizing that the right-hand side in the above equation is a normal variable with variance defined in (4.11). □

Before we state the convergence result for bootstrap, we introduce the notation

$$\widetilde{Y}_N = (\widetilde{\psi}_{0,N}, \ldots, \widetilde{\psi}_{r,N})$$

that represent the bootstrap program (4.7) as an element in space $\mathcal{L}$. Conditioned on $\mathcal{Z}_N$, $\widetilde{Y}_N$ is a random functional w.r.t. $\widetilde{\mathcal{Z}}_b$.

THEOREM 4.3.3 (Converge in bootstrap). *Suppose that Assumption (a)-(c), (e)-(h) holds. Then conditioned on $\mathcal{Z}_N$, with probability one,*

$$n^{1/2}(\gamma(\widetilde{Y}_n)) - \gamma(Y_n)) \xrightarrow{D} N(0, \widetilde{\varsigma}^2), \tag{4.14}$$

here $N(0, \widetilde{\varsigma}^2)$ is a normal variable with mean 0 and variance

$$\widetilde{\varsigma}^2 = E_{\widehat{F}_N(\xi)}[K(\widehat{x}, \widehat{x}_N, \widehat{\lambda}, g, \xi)]^2 - [E_{\widehat{F}_N(\xi)}K(\widehat{x}, \widehat{x}_N, \widehat{\lambda}, g, \xi)]^2, \tag{4.15}$$

with $K(\widehat{x}, \widehat{x}_N, \widehat{\lambda}, g, \xi)$ being a random variable with respect to ξ,

$$K(\widehat{x}, \widehat{x}_N, \widehat{\lambda}, g, \xi) = g_0(\widehat{x}, \xi) - g_0(\widehat{x}_N, \xi) - \sum_i \widehat{\lambda}_i g_i(\widehat{x}_N, \xi).$$

PROOF. By (4.20) in Theorem 4.3.6 and (4.12) , $\gamma(\zeta)$ is Hadamard differentiable at Y_n tangentially to the set $\mathcal{H}$. And for all the vectors ξ in the Bouligand Tangent cone $T_\mu(\mathcal{H})$,

$$\bar{\gamma}'_{Y_N,\mathcal{H}}(\zeta) = \bar{\varphi}(\zeta) - L(\hat{x}_N, \hat{\lambda}, \zeta) \tag{4.16}$$

Now for each i, for some fixed point $x_0 \in S$ as in assumption (f), since $E_\xi |g_i(x_0,\xi)|^2$ is finite under assumption (f)-(g), for N large enough, it follows easily from law of large number that $E_{\hat{F}_N(\xi)}[g_i(x_0,\xi)^2]$ is finite with probability one. Similarly $E_{\hat{F}_N(\xi)}|b_i(\xi)|^2$ is finite almost surely under assumption (g). Therefore, from Thm 4.3.8, conditioned on $\mathcal{Z}_N$, $n^{1/2}(\tilde{Y}_N - Y_N)$ converge to a random element $\widetilde{W}$ in $\mathcal{L}$, and $\widetilde{W}$ follows a vector-valued Gaussian process. That is, for any finite collection $x_1, \ldots, x_m$ in S, $\{\widetilde{W}(x_1), \ldots, \widetilde{W}(x_m)\}$ forms a jointly Gaussian. In particular, we have $\widetilde{W}(\hat{x}_N) = (\tilde{z}_0(\hat{x}_N), \ldots, \tilde{z}_r(\hat{x}_N))$ and $\widetilde{W}(\hat{x}) = (\tilde{z}_0(\hat{x}), \ldots, \tilde{z}_r(\hat{x}))$ being jointly Gaussian.

Therefore using delta method as in [92] (Thm 2.1), we have

$$n^{1/2}(\gamma(\tilde{Y}_N) - \gamma(Y_N)) \xrightarrow{D} \gamma'_{\mu,\mathcal{H}}(\widetilde{W})$$

That is,

$$n^{1/2}(\gamma(\tilde{Y}_N) - \gamma(Y_N)) \xrightarrow{D} \bar{\varphi}(\widetilde{W}) - L(\hat{x}_N, \hat{\lambda}, \widetilde{W}).$$

Equation (4.14) then follows by recognizing that the right-hand side of the above equation is a normal variable conditioned on $\mathcal{Z}_N$.

□

THEOREM 4.3.4. *Suppose Assumptions (a)-(g) stand. Then the bootstrap estimator is strongly consistent in that*

$$\rho_\infty(H_{boot}, H_{\hat{F}_N}) \to 0 \qquad a.s.$$

here $H_{\hat{F}_N}$ is the sample distribution

$$H_{\hat{F}_N}(x) = P\{n^{1/2}(\mathcal{G}(\mathcal{Z}_N) - \mathcal{G}(F)) \leqslant x\}, \tag{4.17}$$

and correspondingly H_{boot} is the bootstrap distribution

$$H_{boot} = P\{n^{1/2}(\mathcal{G}(\tilde{\mathcal{Z}}_b) - \mathcal{G}(\mathcal{Z}_N)) \leqslant x | \mathcal{Z}_N\}. \tag{4.18}$$

PROOF. From Lemma 4.3.6.2 we have $\widehat{x}_N \to x^*$ and $\widehat{\lambda} \to \lambda^*$. Recall that the function

$$K(\widehat{x}, x, \lambda, g, \xi) = g_0(\widehat{x}, \xi) - L(x, \lambda, g) = g_0(\widehat{x}, \xi) - g_0(x, \xi) - \sum_i \lambda_i g_i(x^*, \xi),$$

is continuous in its second and third variable, we have that

$$K(\widehat{x}, \widehat{x}_N, \widehat{\lambda}, g, \xi) \longrightarrow K(\widehat{x}, x^*, \lambda^*, g, \xi).$$

Therefore by Lebesgue dominated convergence theorem, under assumption (f)-(g),

$$\widetilde{\varsigma}^2 / \{E_{\widehat{F}_N}[K(\widehat{x}, x^*, \lambda^*, g, \xi)]^2 - [E_{\widehat{F}_N} K(\widehat{x}, x^*, \lambda^*, g, \xi)]^2\} \to 1$$

Now $\{E_{\widehat{F}_N}[K(\widehat{x}, x^*, \lambda^*, g, \xi)]^2 - [E_{\widehat{F}_N} K(\widehat{x}, x^*, \lambda^*, g, \xi)]^2\}$ can be regarded as the sample variance whose actual variance is ς^2, therefore

$$\frac{E_{\widehat{F}_N}[K(\widehat{x}, x^*, \lambda^*, g, \xi)]^2 - [E_{\widehat{F}_N} K(\widehat{x}, x^*, \lambda^*, g, \xi)]^2}{\varsigma^2} \to 1$$

Combining the two equations yields that

$$\lim \widetilde{\varsigma}^2 = \varsigma^2.$$

The strong consistency of the bootstrap estimator for the optimality gap then follows by recognizing that $n^{1/2}(\mathcal{G}(\mathcal{Z}_N) - \mathcal{G}(G))$ and $n^{1/2}(\mathcal{G}(\widetilde{\mathcal{Z}}_b) - \mathcal{G}(\mathcal{Z}_N))$ have the same limit distribution. □

PROOF OF THEOREM 4.3.1. Theorem 4.3.4 guarantees that the bootstrap estimator is strongly consistent in that

$$\rho_\infty(H_{boot}, H_{\widehat{F}_N}) \to 0 \qquad as.$$

where H_{boot} and $H_{\widehat{F}_N}$ are defined in (4.18) and (4.17) respectively.

Therefore, if we use $\widetilde{\varrho}_{1-\alpha}$ and $\widetilde{\varrho}_\alpha$ to denote the upper and lower quantiles for $\mathcal{G}(\widetilde{\mathcal{Z}}_b) - \mathcal{G}(\mathcal{Z}_N)$, then

$$\lim \inf_{N \to \infty} P(\widetilde{\varrho}_\alpha < \mathcal{G}(\mathcal{Z}_N) - \mathcal{G}(F) \leqslant \widetilde{\varrho}_{1-\alpha}) \geqslant 1 - 2\alpha,$$

Algebraic manipulation yields

$$\liminf_{N\to\infty} P(\widehat{\varrho}_\alpha < \mathcal{G}(F) \leqslant \widehat{\varrho}_{1-\alpha}) \geqslant 1-2\alpha,$$

where $\widehat{\varrho}_{1-\alpha}$ and $\widehat{\varrho}_\alpha$ are the upper and lower quantile for $2\mathcal{G}(\widetilde{\mathcal{Z}}_b) - \mathcal{G}(\mathcal{Z}_N)$.

The inequality (4.8) then follows from the Glivenko–Cantelli theorem which guarantees that the empirical distribution of $2\mathcal{G}(\widetilde{\mathcal{Z}}_b) - \mathcal{G}(\mathcal{Z}_N)$ converges uniformly to the actual distribution of the same variable. □

4.3.4. Auxiliary theorem.

THEOREM 4.3.5. *Suppose that Assumptions (a)-(d) hold. Then the optimal value function $\bar{\phi}(\zeta)$ is Hadamard directionally differentiable at μ tangentially to the set $\mathcal{H}$. And for all the vectors ζ in the Bouligand Tangent cone $T_\mu(\mathcal{H})$,*

$$\bar{\phi}'_{\mu,\mathcal{H}}(\zeta) = L(x^*, \lambda^*, \zeta) \tag{4.19}$$

PROOF. We note that assumption (b) implies that f_i are convex functions that μ is in space $\mathcal{L}$. The theorem is then a corollary from [92], which we include below as Theorem 4.3.7. □

THEOREM 4.3.6. *Suppose that Assumptions (a)-(c) and (h) hold. Then conditioned on $\mathcal{Z}_N$, the optimal value function $\bar{\phi}(\zeta)$ is Hadamard directionally differentiable at Y_N tangentially to the set $\mathcal{H}$. And for all the vectors ζ in the Bouligand Tangent cone $T_\mu(\mathcal{H})$,*

$$\bar{\phi}'_{Y_N,\mathcal{H}}(\zeta) = L(\widehat{x}_N, \widehat{\lambda}, \zeta) \tag{4.20}$$

PROOF. We note that the convexity of g_i implies the convexity of $\psi_{i,N}$, and in turn implies that Y_n is in space $\mathcal{L}$, therefore the approximated problem (4.5) is convex.

Now suppose that $v \in S$ is the point satisfying $f_i(v) < -\epsilon$ for some fixed ϵ. The existence of such a point is guaranteed by the Slater condition in assumption (c). By the Law of Large Numbers, for each $i = 1, \ldots, r$, we have $\phi_{i,N}(v) \to f_i(v)$ a.s. As a result, $\phi_{i,N}(v) < -\frac{\epsilon}{2}$ a.s. for N large enough, so the Slater condition stands for the approximate problem (4.5) a.s.

The statement in the theorem then follows from Thm 4.3.7.

□

LEMMA 4.3.6.1. *Denote*

$$f(x) = \begin{cases} f_0(x), & \text{if } f_i(x) \leqslant 0, i = 0, \ldots, r \\ \infty & \text{otherwise} \end{cases}$$

And

$$\psi_N(x) = \begin{cases} \psi_{0,N}(x), & \text{if } \psi_{i,N}(x) \leqslant 0, i = 0, \ldots, r \\ \infty & \text{otherwise} \end{cases}$$

Suppose Assumptions (a)-(g) stand. Then for any fixed $q \in S$,

$$\liminf_N \psi_N(x_N) \geqslant f(x), \qquad \text{for any sequence } x_N \to x \tag{4.21}$$

and

$$\limsup_N \psi_N(x_N) \leqslant f(x), \qquad \text{for some sequence } x_N \to x \tag{4.22}$$

PROOF. By the law of large numbers, $\psi_{i,N}(x)$ converges point-wise to $f_i(x)$. Under assumptions (a) and (g), $\psi_{i,N}(x)$ is a Lipschitz function defined on a compact set; hence, pointwise convergence implies uniform convergence.

Suppose that β is a cumulative point of $\psi_N(x_N)$. That is, there exists a subsequence $\{N_m\}_{m=1}^{\infty}$, such that $\lim \psi_{N_m}(x_{N_m}) = \beta$. If $\beta = \infty$, then the inequality (4.21) is trivially valid. Otherwise, for each i, we have $\psi_{i,N_m}(x_{N_m}) \leqslant 0$. Since $x_{N_m} \to x$ and ψ_{i,N_m} converge uniformly to f_i, we have $f_i(x) \leqslant 0$, so that $f(x) < \infty$. Furthermore, from the uniform convergence of ψ_{0,N_m} we also have $f(x) = \beta$. This yields

$$\liminf_N \psi_N(x_N) \geqslant f(x)$$

Now, if $f(x) = \infty$, then inequality (4.22) trivially stands. Otherwise, suppose that $f_i(x) \leqslant 0$ for each i. Under assumptions (b) and (c), there exists a point v satisfying $f_i(v) \leqslant -\epsilon < 0$ for some fixed ϵ for $i = 1, \ldots, r$. Define

$$x_N = (1 - a_n)x + a_n v$$

By the convexity of f_i, we have

$$f_i(x_N) \leqslant (1 - a_n)f_i(x) + a_n f_i(v) \leqslant -a_n \epsilon < 0 \qquad i = 1, \ldots, r$$

Since $\psi_{i,N}$ converges uniformly to f_i, there must exists a sequence $a_n \to 0^+$, such that $x_N \to x$ and $\psi_{i,N}(x_N) \leqslant 0$ for each i for N large enough. Then $\psi_N(x_N) = \psi_{0,N}(x_N)$ is a bounded sequence on a compact set, so there must exist a subsequence N_m, such that $\lim \psi_{N_m}(x_{N_m}) = \beta < \infty$. The uniform convergence of ψ_{0,N_m} and the fact that $f_i(x_{N_m}) < 0$ then implies that $f_i(x) = \beta$. Hence (4.22) stands by choosing q_{N_m} as the sequence.

□

LEMMA 4.3.6.2. *Suppose that Assumptions (a)-(g) hold. Then*

$$\widehat{x}_N \to x^*,$$

and

$$\widehat{\lambda} \to \lambda^*$$

where $\widehat{x}_N$ is the optimal solution for (4.5) and $\widehat{\lambda}$ is the Lagrange multiplier associated with $\widehat{x}_N$.

PROOF. By Lemma 4.3.6.1, ψ epi-converge to f. Since the epi-convergence of the functions implies the convergence of the minimizers ([85] Theorem 7.33), we have $\widehat{x}_N \to x^*$. The convergence of $\widehat{\lambda}$ comes from the duality.

□

For reference, we restate some existing results.

THEOREM 4.3.7 ([92] Thm 3.4). *Suppose that the program (4.3) is convex, S is compact, and the Slater condition holds. Then the optimal value function $\bar{\phi}(\zeta)$ is Hadamard directionally differentiable at $\mu = (f_0, \ldots, f_r)$ tangentially to the set $\mathcal{H}$. And for all the vectors ζ in the Bouligand Tangent cone $T_\mu(\mathcal{H})$,*

$$\bar{\phi}'_{\mu,\mathcal{H}}(\zeta) = \min_{x \in S^*(\mu)} \max_{\lambda \in \Lambda(\mu)} L(x, \lambda, \zeta),$$

where $S^*(\mu)$ is the set of optimal solutions of (4.3) and $\Lambda(\mu)$ is the set of Lagrange multipliers satisfying the optimality conditions.

The next theorem requires these assumptions:

(1) The function $g : S \times U \to \mathbb{R}^m$ is continuous in the first variable and measurable in the second.

(2) S is a compact set.

(3) There is a point $x \in S$, such that $E|g(x,d)|^2 < \infty$.

(4) There is a function $b : U \to \mathbb{R}$ with $E|b(d)|^2 < \infty$, such that for all $x, y \in S$,

$$g(x,d) - g(y,d) \leqslant b(d)\|x - y\|.$$

THEOREM 4.3.8 ([53] A3). *Suppose that the function $g : S \times U \to \mathbb{R}^m$ satisfies the above four conditions. Then there exists a Gaussian random variable w taking values in the space of continuous functions $C_m(S)$, such that for any sequence $\{s_i\}$ with i.i.d. random variables,*

$$\sqrt{n}\left[\frac{1}{n}\sum_{j=1}^{n} g(x,s_i) - Eg(x,s_1)\right] \xrightarrow{d} w$$

4.4. Smoothed Point Estimator

In the classical bootstrap method, the confidence interval is constructed around the point estimator $\mathcal{G}(\mathcal{Z}_N)$, which represents the optimality gap under the set $\mathcal{Z}_N$. This is based on the idea that if one does not have access to the entire population F, and instead all that is available is a sample $\mathcal{Z}_N$, then the optimality gap associated with $\mathcal{Z}_N$ serves as a natural surrogate for estimating $\mathcal{G}(F)$. In contrast, the bagging method employs a different approach by generating multiple estimators, each from a distinct subset of $\mathcal{Z}_N$, sampled with or without replacement. The final estimator for the optimality gap is the average of these estimators, which helps to reduce variance and increase stability.

For many important applications, using only the optimality gap associated with the empirical distribution of $\mathcal{Z}_N$ as a point estimator may be insufficient, especially when the sample size is small. In this case, one may seek to use density estimation or probability distribution fitting tools to get a better estimation of the optimality gap. Kernel density estimation and epi-spline fitting [86] are two of the possible tools. The idea of using a smoothed estimator is also proposed in [48, 76] in a different setting.

We describe a general procedure for finding a point estimator, $\bar{G}$, for $\mathcal{G}(F)$ in Algorithm 11. Instead of $\mathcal{G}(\mathcal{Z}_N)$, one may use $\bar{G}$ from Algorithm 11 as a point estimator for the optimality gap. For the form of fitted distribution, we assume that when given enough data points, the fitted distribution should be able to recover the true distribution.

Algorithm 11 Distribution-based Point Estimator

Input: A sample $\mathcal{Z}_N$, replication R, sample size n_c, form of distribution $\breve{F}$.
Fit a distribution function $\breve{F}_N$ using the set $\mathcal{Z}_N$.
for $b \leftarrow 1$ **to** R **do**
    Sample from the distribution $\breve{F}_N$ to get n_c samples $\{\tilde{z}_1^b, \ldots, \tilde{z}_{n_c}^b\}$.
    Compute $G_b = \frac{1}{n_c}\sum_{i=1}^{n_c} h(\widehat{x}, \tilde{z}_i^b) - \min_x \frac{1}{n_c}\sum_{i=1}^{n_c} h(x, \tilde{z}_i^b)$.
Return $\overline{G} = \frac{1}{R}\sum_{b=1}^{R} G_b$ as a point estimator for the optimality gap $\mathcal{G}(F)$.

The above algorithm to some extent provides a unified framework for estimating the optimality gap for stochastic programming problems. We highlight two special cases that link Algorithm 11 back to the point estimators that have been used in the literature.

- Classical Bootstrap: If we do not incorporate smoothness into our form of distribution, but instead use the empirical distribution of $\mathcal{Z}_N$ as the fitted distribution, which assigns an atom of probability with mass $1/N$ to each observation $\mathfrak{z}_i$, then sampling from the distribution $\breve{F}_N$ is equivalent to resampling from the data set $\mathcal{Z}_N$. In this case, with one replication $R = 1$ and $n_c = N$ as the sample size, Algorithm 11 returns $\mathcal{G}(\mathcal{Z}_N)$ as the point estimator, which is the center of the confidence interval in the classical basic bootstrap.
- Classical Bagging: If we use the empirical distribution of $\mathcal{Z}_N$ as $\breve{F}_N$, but run multiple replications ($R > 1$) with possibly smaller resample size n_c, then the returned $\bar{G}$ is a bagging estimator of the optimality gap. The bagging estimator is used as the center of the confidence interval in [59, 60], where a confidence interval is constructed around the optimality gap in stochastic programming via bagging.

4.4.1. Point Estimator Comparisons. We present box plots that illustrate the performance differences among various point estimators used to approximate the optimality gaps using Algorithm 11 and compare these with the true optimality gap. The example problems utilized are detailed in Section 2.5.1.

To simplify the presentation and enhance clarity, we use abbreviations for the estimators. The BT estimator denotes the point estimator derived from the classical bootstrap method. This estimator is constructed using a single batch that encompasses the entire sample set, $\mathcal{Z}_N$. The BG estimator represents the point estimator from the classical bagging method. This estimator is constructed using multiple batches, where each batch is sampled from the empirical distribution of

$\mathcal{Z}_N$. The prefix "S" indicates the application of the smoothed fitted distribution. Therefore, SBT and SBG denote the smoothed versions of the BT and BG estimators, respectively.

The comparative performance of these estimators is as shown in Figures 4.1, 4.2, and 4.3.

While in the CVaR example in Figure 4.1, all of the point estimators tested exhibited a relatively significant error when compared to the actual scale of the true optimality gap, in the farmer example in Figure 4.2 and the multi-knapsack problem in Figure 4.3, the height of the box plots, which represents the error in the point estimators, decreased as we introduce smoothness into the point estimator. This reduction in variance suggests that these point estimators may be more effective in approximating the optimality gap in these situations.

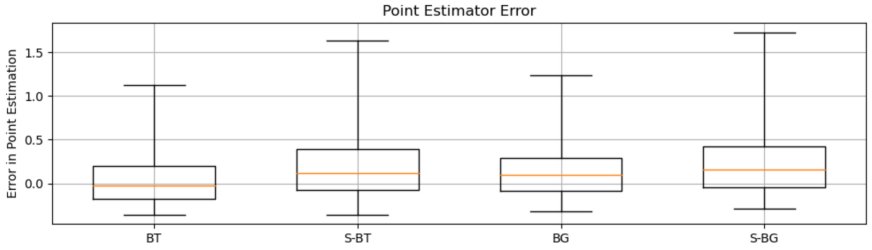

FIGURE 4.1. Results for CVaR problem based on 500 replications. The box-plot displays the distribution of errors in the point estimators with respect to the actual optimality gap 0.36. The size of the set $\mathcal{Z}_N$ is fixed at $N = 40$. The BT point estimator is derived by directly computing the gap associated with the set $\mathcal{Z}_N$. The $S\text{-}BT$ estimator is estimated by resampling $n_c = 8N$ data points from the fitted distribution. For BG and $S\text{-}BG$, we use $R = 400$ replications and the resample size is $n_c = 30$.

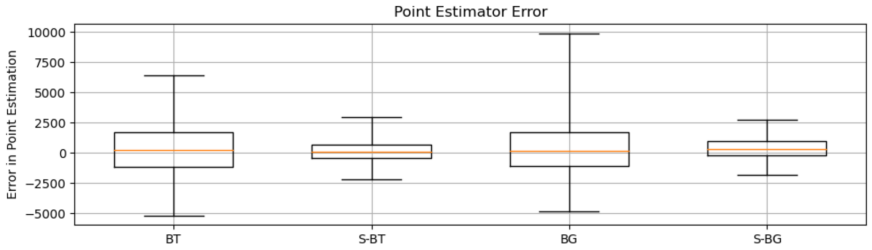

FIGURE 4.2. Results for farmer problem based on 500 replications. The box-plot displays the distribution of errors in the point estimators with respect to the actual optimality gap 7648.32. The size of the set $\mathcal{Z}_N$ is fixed at $N = 40$. The BT point estimator is derived by directly computing the gap associated with the set $\mathcal{Z}_N$. The $S\text{-}BT$ estimator is estimated by resampling $n_c = 8N$ data points from the fitted distribution. For BG and $S\text{-}BG$, we use $R = 400$ replications and the resample size is $n_c = 30$.

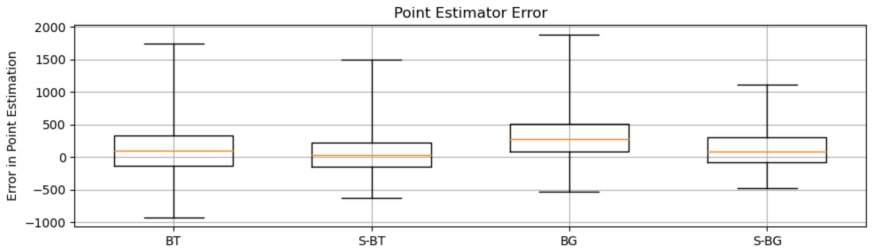

FIGURE 4.3. Results for multi-knapsack problem based on 500 replications. The box-plot displays the distribution of errors in the point estimators with respect to the actual optimality gap 2301.95. The size of the set $\mathcal{Z}_N$ is fixed at $N = 40$. The BT point estimator is derived by directly computing the gap associated with the set $\mathcal{Z}_N$. The $S\text{-}BT$ estimator is estimated by resampling $n_c = 8N$ data points from the fitted distribution. For BG and $S\text{-}BG$, we use $R = 400$ replications and the resample size is $n_c = 30$.

4.5. Smoothed Bootstrap and Smoothed Bagging

In this section, we discuss the situations where we use a smooth function as the fitted distribution function $\breve{F}_N$, and how the existing methods can be adapted to construct a confidence interval around the smoothed point estimator for the optimality gap.

4.5.1. Smoothed Bootstrap. Instead of directly computing a point estimator $\mathcal{G}(\mathcal{Z}_N)$ using the dataset $\mathcal{Z}_N$ as in classical bootstrap, an alternative approach is to employ a smoothed density estimate, such as kernel density estimation, as depicted in Algorithm 11. This algorithm fits a smoothed distribution $\breve{F}_N$ based on $\mathcal{Z}_N$. Subsequently, a large batch of samples is drawn from $\breve{F}_N$ to obtain an estimation. In this case, the estimated center $\bar{G}$ serves as a reliable approximation for $\mathcal{G}(\breve{F}_N)$, which, in turn, provides an estimation for $\mathcal{G}(F)$.

Now, of course, the next question becomes how to construct a confidence interval around $\bar{G}$. The most obvious way would be to directly apply the classical Algorithm 7 . That is, we use $\bar{G}$ in place of $\mathcal{G}(\mathcal{Z}_N)$ in Algorithm 7, and the corresponding return should be a CI for the optimality gap $\mathcal{G}(F)$.

However, if we use a smoothed function for fitting the distribution function $\breve{F}_N$, then it is natural to introduce some smoothness to the bootstrap procedure as well, as in the practice of the standard smoothed bootstrap. That is, instead of resampling from the empirical distribution of $\mathcal{Z}_N$ in Algorithm 7, we instead resample from $\breve{F}_N$, a smoothed version of the empirical c.d.f.,

for bootstrap samples. We use the same smoothed distribution $\breve{F}_N$ for finding the center and for estimating the confidence interval.

We describe in Algorithm 12 the procedure to find a CI for the optimality gap in conjunction with the smoothed point estimator returned by Algorithm 11. We use a standard normal confidence interval in our algorithm and estimate the width of the confidence interval by estimating the variance of the limit distribution, but the percentile bootstrap interval or the bias-corrected and accelerated bootstrap interval ([25]) could also be used here.

Algorithm 12 Smoothed Bootstrap

1: **Input:** A sample set$\mathcal{Z}_N$, number of batches B, form of distribution $\breve{F}$, and a candidate solution $\widehat{x}$
2: **Output:** Confidence interval on the optimality gap
3: Fit a smoothed distribution function $\breve{F}_N$ using the set $\mathcal{Z}_N$.
4: Run Algorithm 11 with the same $\breve{F}_N$, $R = 1$ and a sufficiently large resample size n_c to find the point estimator $\overline{G}$.
5: **for** $b \leftarrow 1$ **to** B **do**
6: Sample from distribution $\breve{F}_N$ to get a new set $\widetilde{\mathcal{Z}}_N^b = \{\tilde{z}_1^b, \ldots, \tilde{z}_N^b\}$.
7: Compute $\mathcal{G}(\widetilde{\mathcal{Z}}_N^b) = \frac{1}{N}\sum_{i=1}^N h(\widehat{x}, \tilde{z}_i^b) - \min_x \frac{1}{N}\sum_{i=1}^N h(x, \tilde{z}_i^b)$,
8: Compute the sample variance s^2 for $\{\mathcal{G}(\widetilde{\mathcal{Z}}_N^b)\}$.
9: Return $[\overline{G} - z_{1-\alpha}s, \overline{G} + z_{1-\alpha}s]$ as the confidence interval (CI).

Algorithm 12 is a small generalization of the classical smoothed bootstrap method, in that we allow different options for the point estimator that serves as the center of the confidence interval. When $\bar{G}$ equals $\mathcal{G}(\mathcal{Z}_N)$, that is, when the empirical distribution is used in Algorithm 11 to provide a point estimator, Algorithm 12 is identical to the classical smoothed bootstrap [30].

Since $\bar{G}$ can essentially be regarded as a smoothed version of $\mathcal{G}(\mathcal{Z}_N)$, the same theory that is used to support the classical smoothed bootstrap method can be used here to justify the asymptotic consistency of the output of Algorithm 7. That is, conditioned on $Z_N = \mathcal{Z}_N$, let $\widetilde{Z}_N^b$ be the random set whose elements obey the distribution $\breve{F}_N$, the distribution of $\mathcal{G}(\widetilde{Z}_N^b) - \bar{G}$ is asymptotically similar to the distribution of $\bar{G} - \mathcal{G}(F)$([23]). Notice that $\bar{G}$ depends on $\breve{F}_N$ and hence in turn is correlated to Z_N.

4.5.2. Smoothed Bagging. The first few steps of the bagging algorithms conform with Algorithm 11 for finding the center of the confidence interval as outlined in Section 4.4. The remaining steps aim to construct an empirical version of the infinitesimal jackknife estimator of the variance, which in turn guides the construction of the confidence interval.

Following the same argument as in the smoothed bootstrap, one may wish to introduce some smoothness into the bagging estimator, especially when the sample size is small. So instead of resampling from the set $\mathcal{Z}_N$ to obtain the bagging sets, we resample from a fitted distribution $\breve{F}_N$ to get the samples and compute the gaps, and take the average of the gaps as our point estimator.

As the infinitesimal jackknife estimator of the variance for the bagging estimator in Algorithm 10 does not directly apply to a smoothed bagging estimator, we seek an alternative approach to estimate the variance using the results from [69]. In particular, consider $\mathcal{G}_k$ as a kernel function $\mathcal{G}_k(\mathfrak{z}_1, \ldots, \mathfrak{z}_k)$ that returns the optimality gap associated with scenarios $\{\mathfrak{z}_1, \ldots, \mathfrak{z}_k\}$, with

$$\mathcal{G}_k(\mathfrak{z}_1, \ldots, \mathfrak{z}_k) = \frac{1}{k}\sum_{i=1}^{k} h(\widehat{x}, \mathfrak{z}_i) - \min_x \frac{1}{k}\sum_{i=1}^{k} h(x, \mathfrak{z}_i)$$

and let

$$\varsigma_{c,k} = var\left(E[\mathcal{G}_k(\mathfrak{z}_1, \ldots, \mathfrak{z}_k)|\mathfrak{z}_1 = \mathfrak{z}_1, \ldots, \mathfrak{z}_c = \mathfrak{z}_c]\right),$$

which is actually the covariance between two instances of the function h with c shared arguments.

By [69, Theorem 1], the variance of the bagging estimator can be estimated with

$$\sigma^2 = \frac{1}{B}\left(\frac{k^2 B}{n}\varsigma_{1,k} + \varsigma_{k,k}\right),$$

with B being the number of bagging sets used to construct the point estimator.

The two variances $\varsigma_{1,k}$ and $\varsigma_{k,k}$ can be estimated using similar Monte Carlo methods as depicted in [69, Section 3]. To estimate $\varsigma_{1,k}$, we start by randomly selecting one scenario $\widetilde{\mathfrak{z}}^{(1)}$ and fixing $\mathfrak{z}_1 = \widetilde{\mathfrak{z}}^{(1)}$, then choose B_{MC} bagging sets $\{\widetilde{\mathcal{Z}}_N^{b'}\}_{b'=1}^{B_{MC}}$, each of size k and contains the fixed seed scenario $\widetilde{\mathfrak{z}}^{(1)}$. Because each bagging set corresponds to an associated optimality gap, the average of the B_{MC} gaps serves as a Monte Carlo approximation to the mean

$$E[\mathcal{G}_k(\mathfrak{z}_1, \ldots, \mathfrak{z}_k)|\mathfrak{z}_1 = \widetilde{\mathfrak{z}}^{(1)}] \approx \bar{G}^{(1)} \overset{def}{=} \frac{1}{B_{MC}}\sum_{b'=1}^{B_{MC}} \mathcal{G}_k(\widetilde{\mathcal{Z}}_N^{b'}).$$

We then repeat the above steps for B_I times, each time with an independently selected fixed, seed scenario $\widetilde{\mathfrak{z}}^{(i)}$. The sample variance among the B_I averaged gaps $\bar{G}^{(i)}$ can then be used as the estimator for $\varsigma_{1,k}$,

$$\varsigma_{1,k} \approx var(\bar{G}^{(1)}, \ldots, \bar{G}^{(B_I)})$$

The variance $\varsigma_{k,k}$ can be estimated using a similar approach, by independently sampling B bagging sets, and computing the variance of the corresponding bagging gaps. But instead of computing the point estimator G and the variances $\varsigma_{1,k}$ and $\varsigma_{k,k}$ in three separate runs, we can incorporate the three procedures into one by utilizing the $B_I * B_{MC}$ gaps (each fixed initial seed scenario yields B_{MC} gaps, and we have in total B_I initial fixed seed scenarios) that are used to estimate $\varsigma_{1,k}$. That is, after generating $B = B_I * B_{MC}$ bagging sets and finding an estimate for the variance $\varsigma_{1,k}$, we take the average of those B gaps

$$G = \frac{1}{B_I * B_{MC}} \sum \mathcal{G}_k(\tilde{\mathcal{Z}}_N^b) = \frac{1}{B_I} \sum_{b=1}^{B_I} \bar{G}^{(b)}$$

to be our bagging estimator G, and use the sample variance of the B gaps as an estimation for $\varsigma_{k,k}$. Algorithm 13 outlines the procedure for constructing a confidence interval around the smoothed bagging estimator. The sample variance s_1 is used to estimate $\varsigma_{1,k}$, and s_2 is for $\varsigma_{k,k}$.

Algorithm 13 Smoothed Bagging with Variance Estimation

1: **Input:** A sample $\mathcal{Z}_N$, number of initial seed points B_I, number of Monte Carlo simulations for each initial point B_{MC}, subsample size k, significance level α, and a candidate solution $\hat{x}$
2: **Output:** Confidence interval on the optimality gap
3: Fit a smoothed distribution function $\breve{F}_N$ using the set $\mathcal{Z}_N$.
4: **for** $b \leftarrow 1$ **to** B_I **do**
5: Select initial seed point $\tilde{z}^{(b)}$ by sampling from $\breve{F}_N$.
6: **for** $b' \leftarrow 1$ **to** B_{MC} **do**
7: Resample from $\breve{F}_N$ to get a bagging set of size k, $\tilde{\mathcal{Z}}_k^{b,b'} = \{\tilde{z}_1^{b,b'}, \ldots, \tilde{z}_k^{b,b'}\}$, including the initial seed point $\tilde{z}^{(b)}$.
8: Compute $\mathcal{G}(\tilde{\mathcal{Z}}_k^{b,b'}) = \frac{1}{k}\sum_{i=1}^{k} h(\hat{x}, \tilde{z}_i^{b,b'}) - \min_x \frac{1}{k}\sum_{i=1}^{k} h(x, \tilde{z}_i^{b,b'})$.
9: Compute the average of the B_{MC} gaps, denoted as $\overline{G}^{(b)}$.
10: Compute the mean of $\overline{G}^{(b)}$ as the center of the confidence interval:

$$G = \frac{1}{B_I} \sum_{b=1}^{B_I} \overline{G}^{(b)} = \frac{1}{B_I \cdot B_{MC}} \sum \mathcal{G}(\tilde{\mathcal{Z}}_k^{b,b'})$$

11: Compute the variance s_1^2 for the B_I averages $\overline{G}^{(b)}$.
12: Compute the variance s_2^2 for all $\mathcal{G}(\tilde{\mathcal{Z}}_k^{b,b'})$.
13: Compute combined variance s^2:

$$s^2 = \frac{1}{B_I \cdot B_{MC}} \left(\frac{k^2 \cdot B_I \cdot B_{MC}}{N} s_1^2 + s_2^2 \right),$$

14: Return $[G - z_{1-\alpha/2}s, G + z_{1-\alpha/2}s]$ as the $(1-\alpha)$ CI for the optimality gap $\mathcal{G}(F)$, with $z_{1-\alpha/2}$ being the $(1-\alpha/2)$ quantile for a standard normal variable.

CHAPTER 5

Uncertainty Quantification in Optimization: Software Implementations

5.1. BOOT-SP: Software for data-based stochastic programming

We developed a Python software, BOOT-SP, for the bootstrap and bagging methods discussed in Chapter 4. The software is accessible via https://github.com/boot-sp/boot-sp.git. The BOOT-SP package is specifically designed to compute point estimators and confidence intervals for the optimality gap in two-stage stochastic programs and is well-suited for conducting simulations tailored to research needs.

The open-source software we describe here takes as input an optimization model and a data sample for the uncertain parameters of the optimization model. A user of the software provides a Python module that has a few procedures, the most important of which takes a sample of data, ξ, as an argument, and returns a Pyomo model for $g(x, \xi)$. The module also contains helper procedures to deal with data processing related to the particular problem. As output, the software provides an estimated solution, a confidence interval for the value of that solution, and a confidence interval for the optimality gap, implying a confidence interval for the optimal objective function value. Hence, it provides a mechanism for data-driven optimization of the expected value, which naturally includes expectation-based risk measures such as CVaR.

The software currently supports the following methods:

- Classical Bootstrap,
- Extended Bootstrap,
- Subsampling,
- Bagging with replacement,
- Bagging without replacement,
- Smoothed Bootstrap, and
- Smoothed Bagging.

For both the Classical and Smoothed Bootstrap methods, the software provides options to generate confidence intervals based on standard normal assumptions or using quantile-based intervals. Additionally, for the Smoothed Bootstrap and Smoothed Bagging methods, users can choose from various forms of fitted distributions, such as kernel density estimation and epi-spline fitting, to tailor the analysis to specific data characteristics and requirements. We are able to exploit the similarities between the methods by creating parameterized functions that perform the calculations.

Figure 5.1 gives an overview of the top-level software components. The function `solve_routine` calls lower-level code in `mpi-sppy` and `Pyomo` to minimize functions of the form $\frac{1}{N}\sum_{\mathfrak{z}_j \in \mathcal{Z}_N} g(\widehat{x}, \mathfrak{z}_j)$. The function `evaluate_scenarios` calls lower-level code in `mpi-sppy` and `Pyomo` to evaluate expectations of the form such as $\frac{1}{N}\sum_{\mathfrak{z}_j \in \mathcal{Z}_N} g(\widehat{x}, \mathfrak{z}_j)$. The differences in methods are due to the way in which these functions are used.

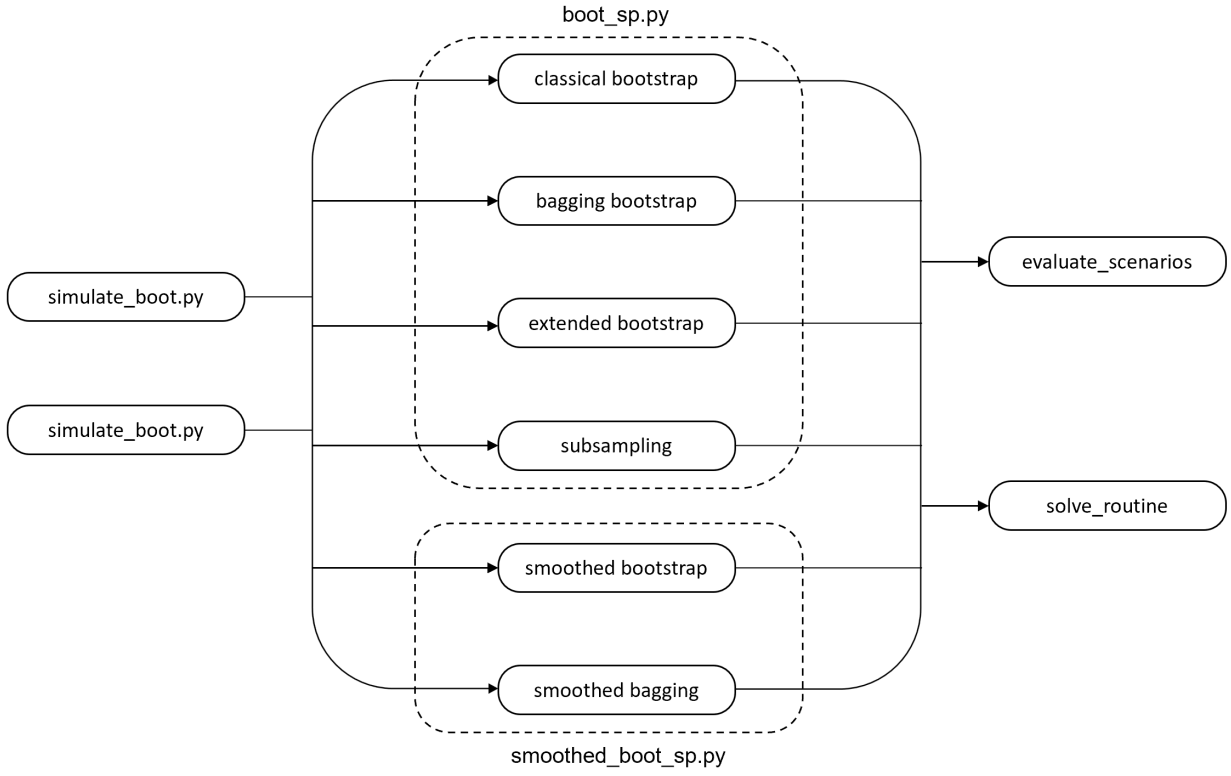

FIGURE 5.1. High level view of software organization.

The algorithms given in Section 4.2 and 4.5 have loops over B. In `boot-sp`, those loops are parallelized using MPI. MPI offers advantages for distributed memory computers at the expense of being difficult to install, particularly on Windows computers.

In most serious applications, we expect that `boot_sp.py` and `smoothed_boot_sp.py` will be used as a callable library. Nonetheless, the `user_boot.py` program is provided to demonstrate the concept of data-driven stochastic programming, so it provides full command line arguments. The main contribution of `boot-sp` is bootstrap or bagging estimation of confidence intervals. Consequently, it is expected the users will have other codes to find a candidate solution $\widehat{x}$. However, in order for `boot-sp` to give a concrete illustration of the concept of data-driving stochastic programming, we offer an option to call a function that uses `mpi-sppy` to compute $\widehat{x}$.

The `simulate_boot.py` software is callable from the command line, but that is mainly for illustration purposes. We also distribute the program `simulate_experiments.py` that runs the experiments in [15]. It calls `simulate_boot.py` in a loop to compute the coverage and width statistics for the confidence intervals.

5.2. Summary Experiment for Smoothed Bootstrap and Smoothed Bagging

We report on simulation experiments to measure the effect of various parameter settings and algorithm design decisions. The relative running times of the algorithms and the quality of their estimates are considered. Out-of-sample tests for the quality of an estimated $\widehat{x}$ are fairly straightforward; however, evaluation of α-level confidence interval estimates of the optimality gap is more complicated because their quality is a two-dimensional object. The dimensions are sometimes called *skill*, which refers to the degree to which $1-\alpha$ of subsequently observed data is within the interval, and *sharpness*, which refers to the size of the interval. The time required to compute the estimates adds a third dimension.

To simplify and enhance the clarity of our experimental outcomes, we employ abbreviations to denote our suggested algorithms. Specifically, we use BT instead of "bootstrap algorithms", and BG instead of "bagging algorithms". We utilize the prefix "S" to signify the application of the smoothed fitted distribution, while the suffix "K/E" will indicate the choice between kernel density estimation and epi-spline fitting. Furthermore, we append the suffix "Q" to BT when employing the quantile method to create bootstrap confidence intervals. Since there were no substantial differences in the results between bagging with replacement and bagging without replacement, the subsequent figures and tables will exclusively display the results obtained using bagging with replacement, which is indicated by the method label BG. Times are given in seconds.

Table 5.1 summarizes the distributions that are used in our algorithms and whether or not the point estimators are obtained via aggregation in each method. The notation F_N is used to represent the empirical distribution derived from the sample, $F_N = \frac{1}{N}\sum \delta_{\mathfrak{z}_i}$, while $\breve{F}_N$ is employed to denote the fitted smoothed distribution function, which can be obtained through methods such as kernel density estimation or epi-spline fitting.

Method	Algorithm	Distribution Used	Aggregated Point Estimator
BT	7	F_N	No
S-BT	12	$\breve{F}_N$	No
BG	10	F_N	Yes
S-BG	13	$\breve{F}_N$	Yes

TABLE 5.1. Method summary

We provide plots that compare the coverage rates and the length of the generated confidence intervals for different methods.

Based on our initial set of experiments, we have observed that, at least for the three examples we have tested, the proposed bagging algorithms provide better point estimators for the center of the confidence intervals when compared with bootstrap methods. Consequently, this leads to a higher coverage rate for the confidence intervals constructed by the bagging method. Within the distinct categories of bootstrap and bagging, we have noticed that when we introduce a "smoothing" effect by incorporating kernel density estimation instead of the empirical distribution for resampling, the coverage rate increases with the length of the confidence interval. See Tables 5.2, 5.3, 5.4.

method	N	B	k	len-avg	len-std	coverage-2	coverage-1	time-avg	time-std
BT	40	400	-	1.05	0.43	0.873	0.890	3.47	0.77
BT-Q	40	400	-	1.01	0.44	0.739	0.751	3.56	0.83
S-BT-K	40	400	-	1.24	0.47	0.907	0.951	9.26	0.33
BG	40	400	20	1.07	0.45	0.900	0.930	1.83	0.40
S-BG	40	10/40	20	1.39	0.61	0.912	0.983	5.40	0.42

TABLE 5.2. Results for CVaR based on 800 replications. coverage-2 reports the coverage rate for the two-sided 90% interval, and coverage-1 reports the coverage rate for the one-sided 95% interval. The center for the smoothed bootstrap method is estimated by resampling $16N$ data points from the fitted distribution. The smoothed-bagging methods used $B_I = 10$ and $B_{MC} = 40$, so that the total number of batches matches the one in the original bagging procedure.

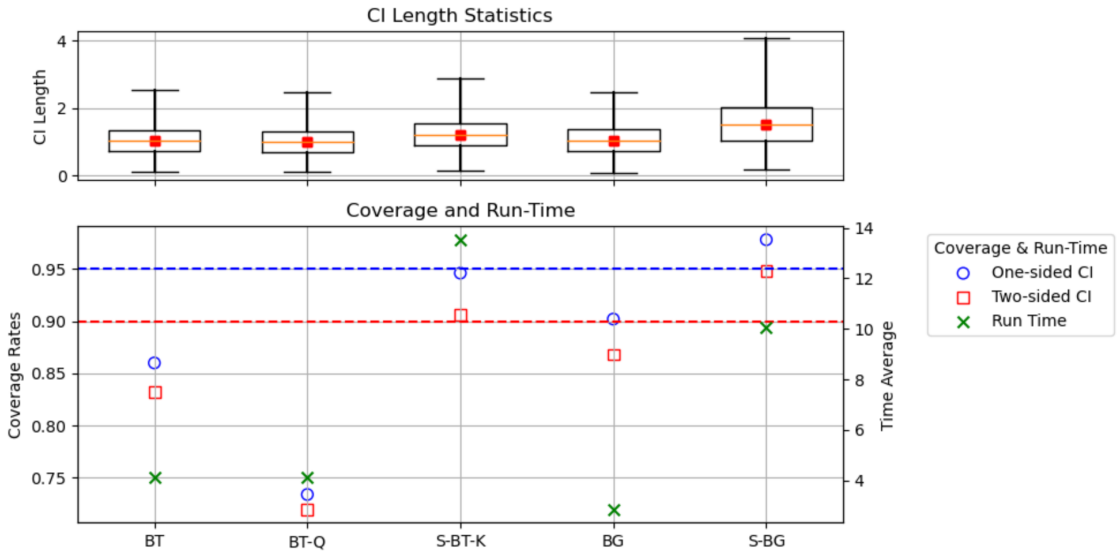

FIGURE 5.2. Results for CVaR based on 500 replications. Two-sided CI reports the coverage rate for the two-sided 90% interval, and One-sided CI reports the coverage rate for the one-sided 95% interval. The center for the smoothed bootstrap method is estimated by resampling $8N$ data points from the fitted distribution. The size of the set $\mathcal{Z}_N$ is fixed at $N = 40$ and the number of bootstrap replications is fixed at $B = 400$. The sub-sample size for the bagging and the smoothed bagging methods is $k = 30$. The smoothed bagging methods used $B_I = 10$ and $B_{MC} = 40$, so that the total number of batches matches the one in the original bagging procedure.

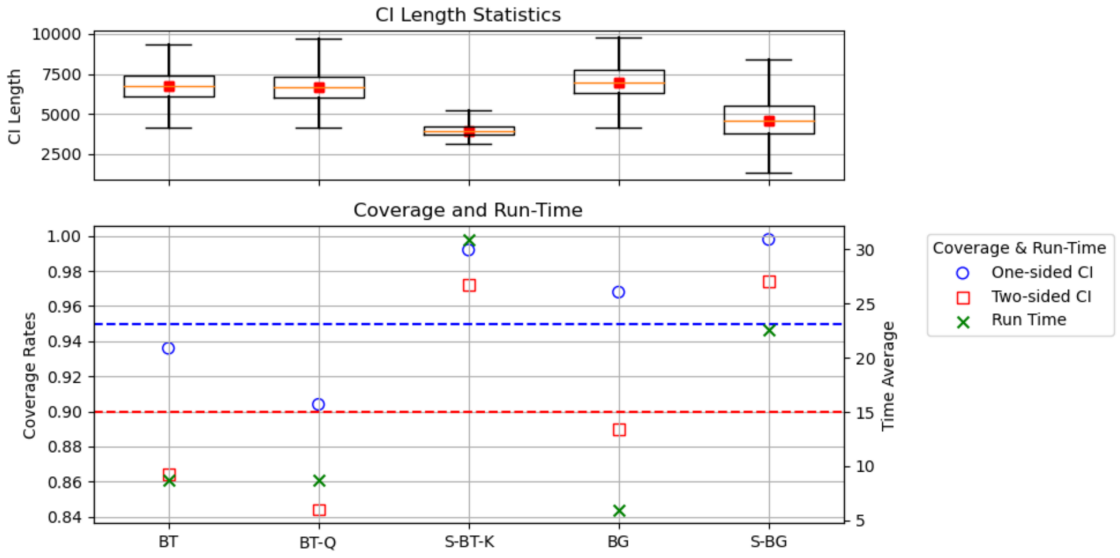

FIGURE 5.3. Results for farmer based on 500 replications. Two-sided CI reports the coverage rate for the two-sided 90% interval, and One-sided CI reports the coverage rate for the one-sided 95% interval. The center for the smoothed bootstrap method is estimated by resampling $8N$ data points from the fitted distribution. The size of the set $\mathcal{Z}_N$ is fixed at $N = 40$ and the number of bootstrap replications is fixed at $B = 400$. The sub-sample size for the bagging and the smoothed bagging methods is $k = 30$. The smoothed bagging methods used $B_I = 10$ and $B_{MC} = 40$, so that the total number of batches matches the one in the original bagging procedure.

method	N	B	k	len-avg	len-std	coverage-2	coverage-1	time-avg	time-std
BT	40	400	-	6776.44	896.65	0.897	0.945	22.23	3.98
BT-Q	40	400	-	6701.22	951.80	0.861	0.910	22.35	3.91
S-BT-K	40	400	-	3972.89	367.00	0.981	0.998	61.40	1.67
BG	40	400	20	6854.71	944.33	0.897	0.988	5.36	1.16
S-BG	40	10/40	20	7063.76	1107.77	1.000	1.000	25.21	1.53

TABLE 5.3. Results for farmer based on 800 replications. coverage-2 reports the coverage rate for the two-sided 90% interval, and coverage-1 reports the coverage rate for the one-sided 95% interval. The center for the smoothed bootstrap method is estimated by resampling $8N$ data points from the fitted distribution. The smoothed-bagging methods used $B_I = 10$ and $B_{MC} = 40$, so that the total number of batches matches the one in the original bagging procedure

method	N	B	k	len-avg	len-std	coverage-2	coverage-1	time-avg	time-std
BT	40	400	-	1243.45	365.85	0.924	0.975	39.47	1.13
BT-Q	40	400	-	1213.02	360.25	0.920	0.940	39.43	1.04
S-BT-K	40	400	-	1618.27	309.96	0.934	0.939	71.61	2.39
BG	40	400	20	1278.33	362.00	0.805	0.998	17.84	0.55
S-BG	40	10/40	20	1718.40	477.98	0.964	0.990	27.64	1.42

TABLE 5.4. Results for multi-knapsack based on 800 replications. coverage-2 reports the coverage rate for the two-sided 90% interval, and coverage-1 reports the coverage rate for the one-sided 95% interval. The center for the smoothed bootstrap method is estimated by resampling $8N$ data points from the fitted distribution. The smoothed-bagging methods used $B_I = 10$ and $B_{MC} = 40$, so that the total number of batches matches the one in the original bagging procedure

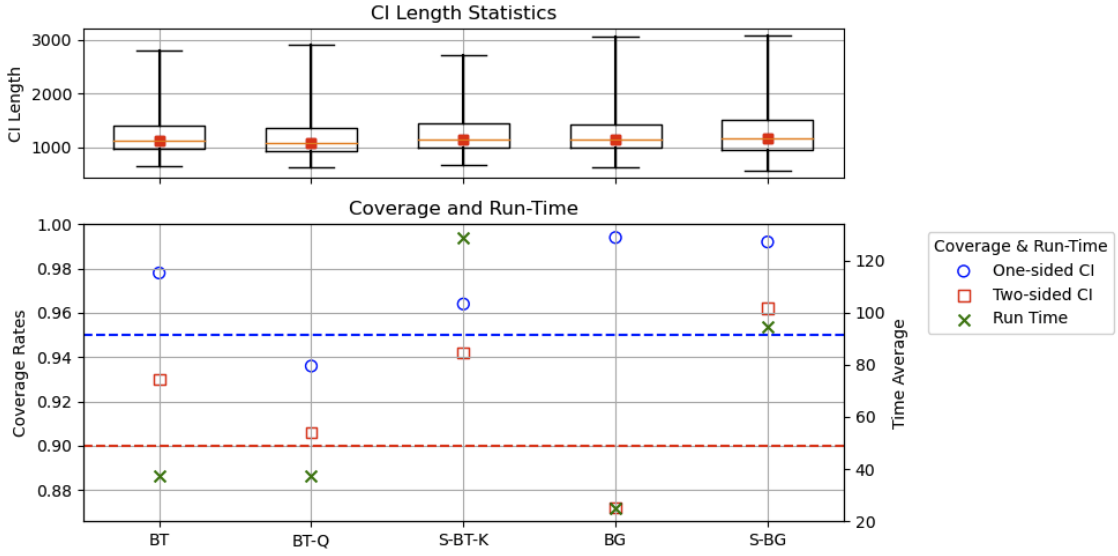

FIGURE 5.4. Results for multi-knapsack problem based on 500 replications. Two-sided CI reports the coverage rate for the two-sided 90% interval, and One-sided CI reports the coverage rate for the one-sided 95% interval. The center for the smoothed bootstrap method is estimated by resampling $8N$ data points from the fitted distribution. The size of the set $\mathcal{Z}_N$ is fixed at $N = 40$ and the number of bootstrap replications is fixed at $B = 400$. The sub-sample size for the bagging and the smoothed bagging methods is $k = 30$. The smoothed bagging methods used $B_I = 10$ and $B_{MC} = 40$, so that the total number of batches matches the one in the original bagging procedure.

5.3. Parameter Selection for Smoothed Bootstrap

In our experiments regarding the parameters of Algorithm 12, we resample n_c points from the fitted smoothed distribution, and use the optimality gap associated with the n_c points as our point

estimator $\bar{G}$. It can be seen from Tables 5.5 and 5.6 that enhanced performance is achieved by resampling more data points from the fitted distribution, as increasing the resample size n_c for the center estimator corresponds to an increase in the coverage rates of the two-sided confidence interval as well.

method	n_c	coverage-2	coverage-1
S-BT-K	320	0.906	0.946
S-BT-K	200	0.880	0.958
S-BT-K	120	0.863	0.965

TABLE 5.5. Results for CVaR with α=0.05 based on 500 replications. The size of the set $\mathcal{Z}_N$ is fixed at $N = 40$, and the number of batches for bootstrap replications is fixed at $B = 400$. Center estimated using varying n_c data points. coverage-2 reports coverage rates for two-sided 90% confidence interval, and coverage-1 reports coverage rates for one-sided 95% confidence interval. The length of the confidence interval remains the same.

method	n_c	coverage-2	coverage-1
S-BT-K	320	0.972	0.992
S-BT-K	200	0.958	0.988
S-BT-K	120	0.932	0.975

TABLE 5.6. Results for farmer with α=0.05 based on 500 replications. The size of the set $\mathcal{Z}_N$ is fixed at $N = 40$, and the number of batches for bootstrap replications is fixed at $B = 400$. Center estimated using varying n_c data points. coverage-2 reports coverage rates for two-sided 90% confidence interval, and coverage-1 reports coverage rates for one-sided 95% confidence interval. The length of the confidence interval remains the same.

For reference, we included the results for the smoothed bootstrap algorithm as in Algorithm 12, but use the empirical point estimator $\mathcal{G}(\mathcal{Z}_N)$ as the point estimator in constructing the confidence interval. As seen in Table 5.7, the confidence interval utilizing the empirical point estimator $\mathcal{G}(\mathcal{Z}_N)$ as the center yields sub-optimal coverage rates and should be avoided in practice.

problem	method	coverage-2	coverage-1
CVaR	S-BT-K	0.703	0.713
farmer	S-BT-K	0.523	0.738

TABLE 5.7. Results with α=0.05 based on 500 replications. The size of the set $\mathcal{Z}_N$ is fixed at $N = 40$, and the number of batches for bootstrap replications is fixed at $B = 400$.

Using the kernel density estimation works well for getting good results with the smoothed bootstrap method, as shown in Table 5.8. It seems to be just as good as or even better than using epi-spline fitting. We noticed that when we switch from epi-spline fitting to kernel density estimation, the coverage rate increases with the same settings. By default, the bandwidth utilized in the kernel density estimation within our code is determined by Scott's rule [90]. However, before finalizing the bandwidth selection, we visually inspect the results of the kernel density estimation to ensure that the resulting curve strikes a balance between smoothness and avoiding over-smoothing. We defer the exploration of the impacts of varying bandwidths to future research endeavors.

problem	method	len-avg	coverage-2	coverage-1
CVaR	S-BT-E	1.02	0.845	0.880
CVaR	S-BT-K	1.23	0.906	0.946
farmer	S-BT-E	3803.31	0.963	0.985
farmer	S-BT-K	3978.63	0.972	0.992

TABLE 5.8. Results with α=0.05 based on 500 replications. The size of the set $\mathcal{Z}_N$ is fixed at $N = 40$, and the number of batches for bootstrap replications is fixed at $B = 400$. The center for the smoothed bootstrap method is estimated by resampling $8N$ data points from the fitted distribution.

5.4. Parameter Selection for Smoothed Bagging

The bagging methods work well in constructing a confidence interval, even when we only have a really small data set at hand. For example, in Table 5.9, with a sample size of $N = 20$, one is still able to apply the bagging methods to construct a confidence interval with a high coverage rate. The introduced smoothness introduces more randomness into the problem, with increased lengths for the confidence intervals and higher coverage rates with a small data set, as can be seen in Table 5.9.

problem	method	B	len-avg	len-std	coverage-2	coverage-1
CVaR	BG	200	1.51	0.83	0.902	0.927
CVaR	S-BG	10/40	1.86	1.08	0.860	0.973
CVaR	S-BG	20/80	1.87	0.91	0.939	0.985
farmer	BG	200	9564.85	1973.87	0.885	0.959
farmer	S-BG	10/40	9907.56	1452.74	1.000	1.000
farmer	S-BG	20/80	9826.73	897.56	1.000	1.000
knapsack	BG	200	1920.37	616.58	0.743	0.998
knapsack	S-BG	10/40	2366.64	697.35	0.887	0.995
knapsack	S-BG	20/80	2451.05	495.75	0.950	0.998

TABLE 5.9. Results based on 800 replications. The size of the set $\mathcal{Z}_N$ is fixed at $N = 20$, and the subsample size $k = N/2$. Coverage-2 reports the coverage rate for the two-sided 90% interval, and coverage-1 reports the coverage rate for the one-sided 95% interval. For the smoothed bagging method, the first number in the B column represents B_I, and the second one is B_{MC}.

Certainly, a larger dataset $\mathcal{Z}_N$ results in an improved estimation of the confidence interval. In Table 5.10, when we increase the sample size from $N = 20$ to $N = 40$, we observe reductions in both the average length and the standard deviation of the interval without sacrificing the superior coverage rate.

method	N	B	len-avg	len-std	coverage-2	coverage-1
BG	20	200	1.51	0.83	0.902	0.927
S-BG	20	20/80	1.87	0.91	0.939	0.985
BG	40	200	1.10	0.47	0.900	0.926
S-BG	40	20/80	1.29	0.52	0.916	0.981

TABLE 5.10. Results for CVaR based on 800 replications. The subsample size k is fixed at $k = N/2$ .Coverage-2 reports the coverage rate for the two-sided 90% interval, and coverage-1 reports the coverage rate for the one-sided 95% interval.

For the smoothed bagging algorithm, both a sufficiently large B_I and a sufficiently large B_{MC} are required to obtain a good coverage without excessive long length in the confidence interval, but as indicated in [69], it is more critical to use a large B_{MC} to obtain an accurate estimation of variance; see Table 5.11.

method	B-I	B-MC	len-avg	len-std	coverage-2	coverage-1
S-BG	10	20	1.69	0.75	0.968	0.994
S-BG	10	100	1.22	0.62	0.879	0.968
S-BG	20	20	1.77	0.67	0.981	0.995
S-BG	20	100	1.30	0.54	0.934	0.980
S-BG	30	20	1.78	0.63	0.983	0.995
S-BG	30	100	1.33	0.51	0.949	0.984

TABLE 5.11. Results for CVaR based on 800 replications. The size of the set $\mathcal{Z}_N$ is fixed at $N = 40$, and the subsample size $k = N/2$. Coverage-2 reports the coverage rate for the two-sided 90% interval, and coverage-1 reports the coverage rate for the one-sided 95% interval. For the smoothed bagging method, the first number in the B column represents B_I, and the second one is B_{MC}.

5.5. Summary of non-smoothed Method Comparisons

We demonstrate experiments on the algorithms over a few examples as detailed below. For simulation purposes, we grant ourselves access to the actual distribution F of the random variable, and approximate the theoretical optimal function value z^* by drawing an extremely large number of samples from the distribution F, then compute the corresponding optimal function value, which we use as z^* for the simulations.

- **Small Schultz Examples**
  - **Unique Solution:** This example is from [89] and used in [31, p. 129]. Results are shown in Table 5.12.
  - **Non-unique Solution:** This example is a modified version of the previous problem used in [31, p. 131]. This problem has multiple optima. Results are shown in Table 5.13.
- **CVaR:** See Section 2.5.1 for the introduction to the CVaR example. Results are shown in Table 5.14.
- **Scalable Farmer:** See Section 2.5.1 for the introduction to the farmer example. Results are shown in Tables 5.15.

We replicate the execution of each algorithm a large number (e.g. 500) times with a fixed candidate solution $\widehat{x}$. In each replication, a new set $\mathcal{Z}_N$ is drawn independently from F, and the algorithm is executed to return a $(1 - 2\alpha)$ confidence interval for the optimal function value z^*. The coverage rate is then reported as the percentages of the confidence intervals that contain z^*

over the 500 runs. We also report the average length of the confidence intervals, denoted as "avg len", over the repeated experiments for each algorithm. The average length of the CIs to some extent represents the sharpness of the intervals computed by the algorithms.

These experiments are intended mainly to illustrate that the software can be used for such experiments. They do illustrate the unsurprising result that if the samples are too small, the confidence intervals will not be very good. They also suggest that the method we call Extended, which is sort of an afterthought in [31], does not seem to work all that well.

The results are mostly reasonable, but mixed and depend on the availability of enough data as well as the choice of method and parameters. Detailed conclusions are beyond the scope of this small study. A preliminary indication is that bagging might be the best thing to try first.

One other thing to note, though, concerns CVaR. Since CVaR considers the tail, getting good confidence intervals requires a larger value of N. Perhaps for similar reasons, quantile-based intervals do not seem to be as good as Gaussian.

method	N	B	k	avg len	coverage
Classical_gaussian	40	100	-	8.04	0.92
Classical_gaussian	80	100	-	5.58	0.90
Classical_gaussian	40	500	-	8.01	0.92
Classical_gaussian	80	500	-	5.73	0.93
Classical_quantile	40	100	-	7.82	0.88
Classical_quantile	80	100	-	5.41	0.87
Classical_quantile	40	500	-	7.97	0.90
Classical_quantile	80	500	-	5.69	0.89
Bagging_with_replacement	40	100	16	9.36	0.93
Bagging_with_replacement	40	100	24	9.38	0.95
Bagging_with_replacement	40	100	32	9.42	0.96
Bagging_with_replacement	80	100	32	7.62	0.97
Bagging_with_replacement	80	100	48	7.39	0.97
Bagging_with_replacement	80	100	64	7.49	0.98
Bagging_with_replacement	40	500	16	8.21	0.90
Bagging_with_replacement	40	500	24	8.15	0.89
Bagging_with_replacement	40	500	32	8.26	0.91
Bagging_with_replacement	80	500	32	6.08	0.93
Bagging_with_replacement	80	500	48	6.05	0.95
Bagging_with_replacement	80	500	64	6.07	0.95
Bagging_without_replacement	40	100	16	9.32	0.95
Bagging_without_replacement	40	100	24	9.59	0.93
Bagging_without_replacement	40	100	32	9.43	0.94
Bagging_without_replacement	80	100	32	7.54	0.98
Bagging_without_replacement	80	100	48	7.83	0.98
Bagging_without_replacement	80	100	64	7.68	0.95
Bagging_without_replacement	40	500	16	8.42	0.91
Bagging_without_replacement	40	500	24	8.58	0.94
Bagging_without_replacement	40	500	32	8.60	0.93
Bagging_without_replacement	80	500	32	6.16	0.93
Bagging_without_replacement	80	500	48	6.29	0.95
Bagging_without_replacement	80	500	64	6.26	0.95
Subsampling	40	100	16	6.15	0.79
Subsampling	40	100	24	5.08	0.73
Subsampling	40	100	32	3.52	0.58
Subsampling	80	100	32	4.36	0.80
Subsampling	80	100	48	3.62	0.72
Subsampling	80	100	64	2.54	0.55
Subsampling	40	500	16	6.30	0.81
Subsampling	40	500	24	5.18	0.74
Subsampling	40	500	32	3.65	0.56
Subsampling	80	500	32	4.45	0.82
Subsampling	80	500	48	3.68	0.74
Subsampling	80	500	64	2.58	0.56
Extended	40	100	-	8.29	0.88
Extended	80	100	-	5.37	0.81
Extended	40	500	-	8.22	0.90
Extended	80	500	-	5.60	0.86

TABLE 5.12. Results for unique_schultz ($z^* \approx$ -62.29) with α=0.1 based on 100 replications.

method	N	B	k	avg len	coverage
Classical_gaussian	40	100	-	7.92	0.90
Classical_gaussian	80	100	-	5.46	0.91
Classical_gaussian	40	500	-	7.89	0.89
Classical_gaussian	80	500	-	5.60	0.93
Classical_quantile	40	100	-	7.72	0.89
Classical_quantile	80	100	-	5.30	0.86
Classical_quantile	40	500	-	7.84	0.90
Classical_quantile	80	500	-	5.57	0.92
Bagging_with_replacement	40	100	16	9.23	0.93
Bagging_with_replacement	40	100	24	9.24	0.94
Bagging_with_replacement	40	100	32	9.28	0.96
Bagging_with_replacement	80	100	32	7.45	0.96
Bagging_with_replacement	80	100	48	7.23	0.97
Bagging_with_replacement	80	100	64	7.32	0.97
Bagging_with_replacement	40	500	16	8.10	0.89
Bagging_with_replacement	40	500	24	8.02	0.89
Bagging_with_replacement	40	500	32	8.13	0.91
Bagging_with_replacement	80	500	32	5.95	0.92
Bagging_with_replacement	80	500	48	5.92	0.94
Bagging_with_replacement	80	500	64	5.94	0.94
Bagging_without_replacement	40	100	16	9.19	0.95
Bagging_without_replacement	40	100	24	9.43	0.93
Bagging_without_replacement	40	100	32	9.27	0.92
Bagging_without_replacement	80	100	32	7.38	0.98
Bagging_without_replacement	80	100	48	7.63	0.98
Bagging_without_replacement	80	100	64	7.50	0.96
Bagging_without_replacement	40	500	16	8.29	0.90
Bagging_without_replacement	40	500	24	8.44	0.91
Bagging_without_replacement	40	500	32	8.46	0.93
Bagging_without_replacement	80	500	32	6.03	0.93
Bagging_without_replacement	80	500	48	6.14	0.96
Bagging_without_replacement	80	500	64	6.11	0.94
Subsampling	40	100	16	6.04	0.82
Subsampling	40	100	24	4.99	0.72
Subsampling	40	100	32	3.45	0.61
Subsampling	80	100	32	4.28	0.81
Subsampling	80	100	48	3.52	0.76
Subsampling	80	100	64	2.48	0.57
Subsampling	40	500	16	6.21	0.83
Subsampling	40	500	24	5.10	0.78
Subsampling	40	500	32	3.59	0.60
Subsampling	80	500	32	4.36	0.83
Subsampling	80	500	48	3.60	0.78
Subsampling	80	500	64	2.52	0.56
Extended	40	100	-	8.16	0.86
Extended	80	100	-	5.16	0.81
Extended	40	500	-	8.08	0.88
Extended	80	500	-	5.45	0.89

TABLE 5.13. Results for nonunique_schultz (z^* ≈ -61.36) with α=0.1 based on 100 replications.

method	N	B	k	avg len	coverage
Classical_gaussian	300	100	-	0.36	0.90
Classical_gaussian	600	100	-	0.26	0.82
Classical_gaussian	300	1000	-	0.36	0.94
Classical_gaussian	600	1000	-	0.26	0.80
Classical_quantile	300	100	-	0.35	0.50
Classical_quantile	600	100	-	0.25	0.57
Classical_quantile	300	1000	-	0.36	0.69
Classical_quantile	600	1000	-	0.26	0.58
Bagging_with_replacement	300	100	120	0.63	1.00
Bagging_with_replacement	300	100	180	0.62	1.00
Bagging_with_replacement	300	100	240	0.62	1.00
Bagging_with_replacement	600	100	240	0.62	1.00
Bagging_with_replacement	600	100	360	0.63	1.00
Bagging_with_replacement	600	100	480	0.62	1.00
Bagging_with_replacement	300	1000	120	0.20	1.00
Bagging_with_replacement	300	1000	180	0.20	1.00
Bagging_with_replacement	300	1000	240	0.20	1.00
Bagging_with_replacement	600	1000	240	0.20	1.00
Bagging_with_replacement	600	1000	360	0.20	1.00
Bagging_with_replacement	600	1000	480	0.20	1.00
Bagging_without_replacement	300	100	120	0.81	1.00
Bagging_without_replacement	300	100	180	0.99	1.00
Bagging_without_replacement	300	100	240	1.39	1.00
Bagging_without_replacement	600	100	240	0.80	1.00
Bagging_without_replacement	600	100	360	0.99	1.00
Bagging_without_replacement	600	100	480	1.39	1.00
Bagging_without_replacement	300	1000	120	0.26	1.00
Bagging_without_replacement	300	1000	180	0.31	1.00
Bagging_without_replacement	300	1000	240	0.45	1.00
Bagging_without_replacement	600	1000	240	0.26	1.00
Bagging_without_replacement	600	1000	360	0.32	1.00
Bagging_without_replacement	600	1000	480	0.45	1.00
Subsampling	300	100	120	0.36	0.65
Subsampling	300	100	180	0.35	0.61
Subsampling	300	100	240	0.35	0.60
Subsampling	600	100	240	0.26	0.64
Subsampling	600	100	360	0.26	0.60
Subsampling	600	100	480	0.25	0.50
Subsampling	300	1000	120	0.36	0.67
Subsampling	300	1000	180	0.36	0.65
Subsampling	300	1000	240	0.36	0.62
Subsampling	600	1000	240	0.26	0.63
Subsampling	600	1000	360	0.26	0.62
Subsampling	600	1000	480	0.26	0.67
Extended	300	100	-	0.50	0.79
Extended	600	100	-	0.36	0.71
Extended	300	1000	-	0.51	0.85
Extended	600	1000	-	0.36	0.81

TABLE 5.14. Results for cvar ($z^* \approx 1.79$) with α=0.1 based on 100 replications.

method	N	B	k	avg len	coverage
Classical_gaussian	30	100	-	28259.46	0.887
Classical_gaussian	60	100	-	20181.52	0.892
Classical_gaussian	30	1000	-	28294.08	0.885
Classical_gaussian	60	1000	-	20272.41	0.907
Classical_quantile	30	100	-	27343.29	0.870
Classical_quantile	60	100	-	19645.95	0.877
Classical_quantile	30	1000	-	28236.30	0.880
Classical_quantile	60	1000	-	20235.36	0.905
Bagging_with_replacement	30	100	12	31422.27	0.932
Bagging_with_replacement	30	100	18	31257.34	0.938
Bagging_with_replacement	30	100	24	32061.31	0.932
Bagging_with_replacement	60	100	24	25164.00	0.955
Bagging_with_replacement	60	100	36	25298.68	0.953
Bagging_with_replacement	60	100	48	25285.28	0.950
Bagging_with_replacement	30	1000	12	28323.03	0.895
Bagging_with_replacement	30	1000	18	28489.61	0.902
Bagging_with_replacement	30	1000	24	28554.10	0.900
Bagging_with_replacement	60	1000	24	20614.13	0.907
Bagging_with_replacement	60	1000	36	20688.71	0.907
Bagging_with_replacement	60	1000	48	20751.51	0.905
Bagging_without_replacement	30	100	12	32638.99	0.935
Bagging_without_replacement	30	100	18	32640.54	0.920
Bagging_without_replacement	30	100	24	32558.19	0.930
Bagging_without_replacement	60	100	24	25546.72	0.965
Bagging_without_replacement	60	100	36	25369.97	0.963
Bagging_without_replacement	60	100	48	25711.17	0.970
Bagging_without_replacement	30	1000	12	29438.81	0.902
Bagging_without_replacement	30	1000	18	29629.62	0.905
Bagging_without_replacement	30	1000	24	29664.24	0.895
Bagging_without_replacement	60	1000	24	21069.04	0.915
Bagging_without_replacement	60	1000	36	21175.03	0.917
Bagging_without_replacement	60	1000	48	21287.06	0.922
Subsampling	30	100	12	21560.28	0.777
Subsampling	30	100	18	17703.34	0.680
Subsampling	30	100	24	12593.48	0.502
Subsampling	60	100	24	15389.09	0.785
Subsampling	60	100	36	12457.75	0.680
Subsampling	60	100	48	8943.31	0.500
Subsampling	30	1000	12	22124.33	0.797
Subsampling	30	1000	18	18209.22	0.690
Subsampling	30	1000	24	12877.64	0.510
Subsampling	60	1000	24	15739.58	0.790
Subsampling	60	1000	36	12903.63	0.700
Subsampling	60	1000	48	9166.89	0.525
Extended	30	100	-	26890.50	0.848
Extended	60	100	-	19620.82	0.863
Extended	30	1000	-	28080.65	0.870
Extended	60	1000	-	20435.31	0.875

TABLE 5.15. Results for farmer (z^* ≈ -132750.32) with α=0.1 based on 400 replications.

CHAPTER 6

Conclusion

The dissertation explored the applications of Monte Carlo and Bootstrap methods to stochastic optimization.

Given a candidate solution for a stochastic optimization problem, we propose a method to effectively compute the function value by constructing importance sampling distributions using surrogate modeling. Our methods strive to reduce the need for evaluation of the recourse function because these evaluations often require the use of optimization algorithms and can be computationally expensive. We show numerical results that demonstrate that the proposed method offers a significant improvement over Monte Carlo sampling even when augmented by importance sampling because the surrogate model allows for much faster evaluation than is required to solve an optimization problem. We also demonstrate good parallel efficiency for up to 16 processors.

Nest, we focus on uncertainty quantification in stochastic optimization when the underlying distribution is known, and we discussed Multiple Replication Procedures and Sequential Sampling Procedures. These procedures are used to create confidence intervals around the optimality gap of a given first-stage solution for two-stage and multi-stage stochastic programs. In addition, we explore bootstrap and bagging methods, which utilize only sampled data to derive asymptotically valid estimated solutions and quality assessments. We introduced various combinations of distribution estimation and resampling techniques for data-driven stochastic programming problems. Specifically, we adapt the smoothed bootstrap method and develop the smoothed bagging method in the context of stochastic optimization. These algorithms are designed to acquire solutions and calculate the confidence intervals for the optimality gap.

Finally, we introduce the software tool BOOT-SP developed for uncertainty quantification in stochastic programming. This tool efficiently generates both a consistent sample-average solution and reliable estimates of confidence intervals for the optimality gap using the discussed bootstrap and bagging techniques. Our numerical experiments demonstrated the effectiveness of the smoothed bootstrap and smoothed bagging methods in constructing confidence intervals for small datasets,

although with a longer computational time compared to empirical bootstrap and bagging algorithms.

Despite the promising results, several questions remain open for further investigation. In terms of evaluating function values using the importance sampling method with surrogate modeling, future research should explore how to incorporate prior knowledge into the construction of the surrogate.

Regarding uncertainty quantification with smoothed bootstrap and smoothed bagging, a primary area of inquiry involves determining which types of problem most benefit from the smoothness effect introduced by the proposed algorithms. These methods present a trade-off between achieving high coverage rates and reducing computational time. Consequently, it becomes interesting to explore the optimal number of bootstrap samples required to yield accurate estimations. An additional experimental question addresses the common challenge of sample allocation: specifically, determining the optimal distribution of finite samples between computing the candidate solution and estimating confidence intervals.

Bibliography

[1] T. Alsup and B. Peherstorfer, *Context-aware surrogate modeling for balancing approximation and sampling costs in multifidelity importance sampling and bayesian inverse problems*, SIAM/ASA Journal on Uncertainty Quantification, 11 (2023), pp. 285–319.

[2] M. Anitescu and C. Petra, *Higher-order confidence intervals for stochastic programming using bootstrapping*, tech. rep., Citeseer, 2011.

[3] A. Azaron, K. Brown, S. A. Tarim, and M. Modarres, *A multi-objective stochastic programming approach for supply chain design considering risk*, International Journal of Production Economics, 116 (2008), pp. 129–138.

[4] T. G. Bailey, P. A. Jensen, and D. P. Morton, *Response surface analysis of two-stage stochastic linear programming with recourse*, Naval Research Logistics (NRL), 46 (1999), pp. 753–776.

[5] G. Bayraksan and D. P. Morton, *Assessing solution quality in stochastic programs*, Mathematical Programming, 108 (2006), pp. 495–514.

[6] ———, *Assessing solution quality in stochastic programs via sampling*, in Decision Technologies and Applications, Informs, 2009, pp. 102–122.

[7] ———, *A sequential sampling procedure for stochastic programming*, Operations Research, 59 (2011), pp. 898–913.

[8] G. Bayraksan and P. Pierre-Louis, *Fixed-width sequential stopping rules for a class of stochastic programs*, SIAM Journal on Optimization, 22 (2012), pp. 1518–1548.

[9] ———, *Fixed-width sequential stopping rules for a class of stochastic programs*, SIAM Journal on Optimization, 22 (2012), pp. 1518–1548.

[10] J. R. Birge and F. Louveaux, *Introduction to stochastic programming*, Springer Science & Business Media, 2011.

[11] L. Breiman, *Bagging predictors*, Machine learning, 24 (1996), pp. 123–140.

[12] P. Bühlmann and B. Yu, *Analyzing bagging*, The annals of Statistics, 30 (2002), pp. 927–961.

[13] M. L. Bynum, G. A. Hackebeil, W. E. Hart, C. D. Laird, B. L. Nicholson, J. D. Siirola, J.-P. Watson, and D. L. Woodruff, *Pyomo–optimization modeling in python*, vol. 67, Springer Science & Business Media, third ed., 2021.

[14] X. Chen, S. Cazaux, B. C. Knight, and D. L. Woodruff, *Confidence interval software for multi-stage stochastic programs*. Optimization Online, 2021.

[15] X. Chen and D. L. Woodruff, *Software for data-based stochastic programming using bootstrap estimation*, INFORMS Journal on Computing, 35 (2023), pp. 1218–1224.

[16] ——, *Distributions and bootstrap for data-based stochastic programming*, Computational Management Science, 21 (2024).

[17] ——, *Importance sampling in optimization under uncertainty using surrogate models*, in Proceedings of the Winter Simulation Conference, 2024. To appear.

[18] A. Chiralaksanakul and D. P. Morton, *Assessing policy quality in multi-stage stochastic programming*, Humboldt-Universität zu Berlin, Mathematisch-Naturwissenschaftliche Fakultät II, Institut für Mathematik, 2004.

[19] G. B. Dantzig, *Linear programming under uncertainty*, Management science, 1 (1955), pp. 197–206.

[20] G. B. Dantzig and P. W. Glynn, *Parallel processors for planning under uncertainty*, Annals of Operations research, 22 (1990), pp. 1–21.

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

[22] A. C. Davison and D. V. Hinkley, *Bootstrap methods and their application*, Cambridge university press, 1997.

[23] D. De Angelis and G. A. Young, *Smoothing the bootstrap*, International Statistical Review/Revue Internationale de Statistique, (1992), pp. 45–56.

[24] V. L. De Matos, D. P. Morton, and E. C. Finardi, *Assessing policy quality in a multistage stochastic program for long-term hydrothermal scheduling*, Annals of Operations Research, 253 (2017), pp. 713–731.

[25] T. J. Diciccio and J. P. Romano, *A review of bootstrap confidence intervals*, Journal of the Royal Statistical Society Series B: Statistical Methodology, 50 (1988), pp. 338–354.

[26] O. Dowson and L. Kapelevich, *Sddp.jl: A julia package for stochastic dual dynamic programming*, INFORMS Journal on Computing, 33 (2020), pp. 27–33.

[27] S. S. Drew and T. Homem-de Mello, *Quas-monte carlo strategies for stochastic optimization*, in Proceedings of the 2006 winter simulation conference, IEEE, 2006, pp. 774–782.

[28] J. Dupacová and R. Wets, *Asymptotic behavior of statistical estimators and of optimal solutions of stochastic optimization problems*, The annals of statistics, 16 (1988), pp. 1517–1549.

[29] B. Efron, *Nonparametric estimates of standard error: the jackknife, the bootstrap and other methods*, Biometrika, 68 (1981), pp. 589–599.

[30] ——, *The jackknife, the bootstrap and other resampling plans*, SIAM, 1982.

[31] A. Eichhorn and W. Römisch, *Stochastic integer programming: Limit theorems and confidence intervals*, Mathematics of Operations Research, 32 (2007), pp. 118–135.

[32] Y. M. Ermoliev and R.-B. Wets, *Numerical techniques for stochastic optimization*, Springer-Verlag, 1988.

[33] M. B. Freimer, J. T. Linderoth, and D. J. Thomas, *The impact of sampling methods on bias and variance in stochastic linear programs*, Computational Optimization and Applications, 51 (2012), pp. 51–75.

[34] D. Frisch and U. D. Hanebeck, *Rejection sampling from arbitrary multivariate distributions using generalized fibonacci lattices*, in 2022 25th International Conference on Information Fusion (FUSION), IEEE, 2022, pp. 1–7.

[35] R. Fuentes and A. Lillo-Bañuls, *Smoothed bootstrap malmquist index based on dea model to compute productivity of tax offices*, Expert systems with applications, 42 (2015), pp. 2442–2450.

[36] R. A. Garrido, P. Lamas, and F. J. Pino, *A stochastic programming approach for floods emergency logistics*, Transportation research part E: logistics and transportation review, 75 (2015), pp. 18–31.

[37] S. S. Garud, I. Karimi, and M. Kraft, *Smart sampling algorithm for surrogate model development*, Computers & Chemical Engineering, 96 (2017), pp. 103–114.

[38] H. Haario, E. Saksman, and J. Tamminen, *An adaptive metropolis algorithm*, Bernoulli, (2001), pp. 223–242.

[39] W. E. Hart, J.-P. Watson, and D. L. Woodruff, *Pyomo: Modeling and solving mathematical programs in Python*, Math. Program. Comput., 3 (2011).

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

[41] J. L. Higle and S. Sen, *Stochastic decomposition: An algorithm for two-stage linear programs with recourse*, Mathematics of operations research, 16 (1991), pp. 650–669.

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

[43] J. L. Higle and S. Sen, *Statistical approximations for stochastic linear programming problems*, Annals of operations research, 85 (1999), pp. 173–193.

[44] ———, *Stochastic decomposition: a statistical method for large scale stochastic linear programming*, vol. 8, Springer Science & Business Media, 2013.

[45] T. Homem-de Mello, *On rates of convergence for stochastic optimization problems under non–independent and identically distributed sampling*, SIAM Journal on Optimization, 19 (2008), pp. 524–551.

[46] T. Homem-de Mello and G. Bayraksan, *Monte carlo sampling-based methods for stochastic optimization*, Surveys in Operations Research and Management Science, 19 (2014), pp. 56–85.

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

[48] W. T. Huh, R. Levi, P. Rusmevichientong, and J. B. Orlin, *Adaptive data-driven inventory control with censored demand based on kaplan-meier estimator*, Operations Research, 59 (2011), pp. 929–941.

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

[50] A. S. Kenyon and D. P. Morton, *Stochastic vehicle routing with random travel times*, Transportation Science, 37 (2003), pp. 69–82.

[51] E. Keyvanshokooh, S. M. Ryan, and E. Kabir, *Hybrid robust and stochastic optimization for closed-loop supply chain network design using accelerated benders decomposition*, European Journal of Operational Research, 249 (2016), pp. 76–92.

[52] A. J. King, *Modeling with stochastic programming*, Springer, 2012.

[53] A. J. King and R. T. Rockafellar, *Asymptotic theory for solutions in statistical estimation and stochastic programming*, Mathematics of Operations Research, 18 (1993), pp. 148–162.

[54] A. J. Kleywegt, A. Shapiro, and T. Homem-de Mello, *The sample average approximation method for stochastic discrete optimization*, SIAM Journal on optimization, 12 (2002), pp. 479–502.

[55] K. A. Klise, B. L. Nicholson, A. Staid, and D. L. Woodruff, *Parmest: Parameter estimation via pyomo*, in Proceedings of the 9th International Conference on Foundations of Computer-Aided Process Design, S. G. Muñoz, C. D. Laird, and M. J. Realff, eds., vol. 47 of Computer Aided Chemical Engineering, Elsevier, 2019, pp. 41–46.

[56] B. Knueven, D. Mildebrath, C. Muir, J. D. Siirola, J.-P. Watson, and D. L. Woodruff, *A parallel hub-and-spoke system for large-scale scenario-based optimization under uncertainty*, https://mpi-sppy.readthedocs.io/en/latest/, (2021).

[57] M. Koivu, *Variance reduction in sample approximations of stochastic programs*, Mathematical programming, 103 (2005), pp. 463–485.

[58] V. Kozmík and D. P. Morton, *Evaluating policies in risk-averse multi-stage stochastic programming*, Mathematical Programming, 152 (2015), pp. 275–300.

[59] H. Lam and H. Qian, *Assessing solution quality in stochastic optimization via bootstrap aggregating*, in 2018 Winter Simulation Conference (WSC), IEEE, 2018, pp. 2061–2071.

[60] ———, *Bounding optimality gap in stochastic optimization via bagging: Statistical efficiency and stability*, arXiv preprint arXiv:1810.02905, (2018).

[61] G. Lan and Z. Zhou, *Algorithms for stochastic optimization with expectation constraints*, arXiv preprint arXiv:1604.03887, (2016).

[62] P. L'Ecuyer and C. Lemieux, *Recent advances in randomized quasi-monte carlo methods*, Modeling uncertainty: An examination of stochastic theory, methods, and applications, (2002), pp. 419–474.

[63] Y. Li and Y.-G. Wang, *Smooth bootstrap methods for analysis of longitudinal data*, Statistics in medicine, 27 (2008), pp. 937–953.

[64] J. Linderoth, A. Shapiro, and S. Wright, *The empirical behavior of sampling methods for stochastic programming*, Annals of Operations Research, 142 (2006), pp. 215–241.

[65] M. Lubin, O. Dowson, J. Dias Garcia, J. Huchette, B. Legat, and J. P. Vielma, *JuMP 1.0: Recent improvements to a modeling language for mathematical optimization*, Mathematical Programming Computation, (2023).

[66] T. J. Mackman, C. B. Allen, M. Ghoreyshi, and K. Badcock, *Comparison of adaptive sampling methods for generation of surrogate aerodynamic models*, AIAA journal, 51 (2013), pp. 797–808.

[67] 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 (1999), pp. 47–56.

[68] M. D. McKay, R. J. Beckman, and W. J. Conover, *A comparison of three methods for selecting values of input variables in the analysis of output from a computer code*, Technometrics, 42 (2000), pp. 55–61.

[69] L. Mentch and G. Hooker, *Quantifying uncertainty in random forests via confidence intervals and hypothesis tests*, The Journal of Machine Learning Research, 17 (2016), pp. 841–881.

[70] A. Narayan, C. Gittelson, and D. Xiu, *A stochastic collocation algorithm with multifidelity models*, SIAM Journal on Scientific Computing, 36 (2014), pp. A495–A521.

[71] B. L. Nelson, *Control variate remedies*, Operations Research, 38 (1990), pp. 974–992.

[72] C. Nentwich and S. Engell, *Surrogate modeling of phase equilibrium calculations using adaptive sampling*, Computers & Chemical Engineering, 126 (2019), pp. 204–217.

[73] H. Niederreiter, *Random number generation and quasi-Monte Carlo methods*, SIAM, 1992.

[74] V. I. Norkin, G. C. Pflug, and A. Ruszczyński, *A branch and bound method for stochastic global optimization*, Mathematical programming, 83 (1998), pp. 425–450.

[75] L. Ntaimo, *Computational Stochastic Programming*, To publish in Springer Optimization and its Applications, 2023.

[76] P. Parpas, B. Ustun, M. Webster, and Q. K. Tran, *Importance sampling in stochastic programming: A markov chain monte carlo approach*, INFORMS Journal on Computing, 27 (2015), pp. 358–377.

[77] B. Peherstorfer, T. Cui, Y. Marzouk, and K. Willcox, *Multifidelity importance sampling*, Computer Methods in Applied Mechanics and Engineering, 300 (2016), pp. 490–509.

[78] B. Peherstorfer, K. Willcox, and M. Gunzburger, *Survey of multifidelity methods in uncertainty propagation, inference, and optimization*, Siam Review, 60 (2018), pp. 550–591.

[79] M. V. Pereira and L. M. Pinto, *Multi-stage stochastic optimization applied to energy planning*, Mathematical programming, 52 (1991), pp. 359–375.

[80] E. L. Plambeck, B.-R. Fu, S. M. Robinson, and R. Suri, *Sample-path optimization of convex stochastic performance functions*, Mathematical Programming, 75 (1996), pp. 137–176.

[81] W. B. Powell and H. Topaloglu, *Stochastic programming in transportation and logistics*, Handbooks in operations research and management science, 10 (2003), pp. 555–635.

[82] A. Prékopa, *Stochastic programming*, vol. 324, Springer Science & Business Media, 2013.

[83] Y. Raviv and N. Intrator, *Bootstrapping with noise: An effective regularization technique*, Connection Science, 8 (1996), pp. 355–372.

[84] S. M. Robinson, *Analysis of sample-path optimization*, Mathematics of Operations Research, 21 (1996), pp. 513–528.

[85] R. T. Rockafellar and R. J.-B. Wets, *Variational analysis*, vol. 317, Springer Science & Business Media, 2009.

[86] J. O. Royset and R. J.-B. Wets, *Fusion of hard and soft information in nonparametric density estimation*, European Journal of Operational Research, 247 (2015), pp. 532–547.

[87] A. Ruszczyński and A. Shapiro, *Stochastic programming models*, Handbooks in operations research and management science, 10 (2003), pp. 1–64.

[88] T. Santoso, S. Ahmed, M. Goetschalckx, and A. Shapiro, *A stochastic programming approach for supply chain network design under uncertainty*, European Journal of Operational Research, 167 (2005), pp. 96–115.

[89] R. Schultz, L. Stougie, and M. H. Van Der Vlerk, *Solving stochastic programs with integer recourse by enumeration: A framework using gröbner basis*, Mathematical Programming, 83 (1998), pp. 229–252.

[90] D. W. Scott, *Multivariate density estimation: theory, practice, and visualization*, John Wiley & Sons, 2015.

[91] J. Shao and D. Tu, *The jackknife and bootstrap*, Springer Science & Business Media, 2012.

[92] A. Shapiro, *Asymptotic analysis of stochastic programs*, Annals of Operations Research, 30 (1991), pp. 169–186.

[93] ———, *Inference of statistical bounds for multistage stochastic programming problems*, Mathematical Methods of Operations Research, 58 (2003), pp. 57–68.

[94] ———, *Statistical inference of multistage stochastic programming problems*, Math. Methods of Oper. Res, 58 (2003), pp. 57–68.

[95] ———, *Analysis of stochastic dual dynamic programming method*, European Journal of Operational Research, 209 (2011), pp. 63–72.

[96] B. Silverman and G. Young, *The bootstrap: to smooth or not to smooth?*, Biometrika, 74 (1987), pp. 469–479.

[97] K. Song, Y. Zhang, X. Zhuang, X. Yu, and B. Song, *Reliability-based design optimization using adaptive surrogate model and importance sampling-based modified sora method*, Engineering with Computers, 37 (2021), pp. 1295–1314.

[98] H. Vaagen and S. W. Wallace, *Product variety arising from hedging in the fashion supply chains*, International Journal of Production Economics, 114 (2008), pp. 431–455.

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

[100] S. W. Wallace and S.-E. Fleten, *Stochastic programming models in energy*, Handbooks in operations research and management science, 10 (2003), pp. 637–677.

[101] N.-C. Xiao, M. J. Zuo, and C. Zhou, *A new adaptive sequential sampling method to construct surrogate models for efficient reliability analysis*, Reliability Engineering & System Safety, 169 (2018), pp. 330–338.

[102] H. Xu, L. Liu, and M. Zhang, *Adaptive surrogate model-based optimization framework applied to battery pack design*, Materials & Design, 195 (2020), p. 108938.

[103] X. Xu and J. R. Birge, *Equity valuation, production, and financial planning: A stochastic programming approach*, Naval Research Logistics (NRL), 53 (2006), pp. 641–655.

[104] J. Yao, Z. Ye, and Y. Wang, *Efficient importance sampling for high-sigma yield analysis with adaptive online surrogate modeling*, in 2013 Design, Automation & Test in Europe Conference & Exhibition (DATE), IEEE, 2013, pp. 1291–1296.

[105] W. T. Ziemba and R. G. Vickson, *Stochastic optimization models in finance*, Academic Press, 2014.