Importance Sampling in Stochastic Programming: A Markov Chain Monte Carlo Approach

Panos Parpas
Department of Computer Science
Imperial College London, South Kensington, SW7 2AZ
p.parpas@imperial.ac.uk

Berk Ustun
Department of Electrical Engineering and Computer Science
Massachusetts Institute of Technology, Cambridge, MA 02139
ustunb@mit.edu

Mort Webster
Engineering Systems Division
Massachusetts Institute of Technology, Cambridge, MA 02139
mort@mit.edu

Quang Kha Tran
Department of Computer Science
Imperial College London, South Kensington, SW7 2AZ
quang.tran07@imperial.ac.uk

Stochastic programming models are large-scale optimization problems that are used to facilitate decision-making under uncertainty. Optimization algorithms for such problems need to evaluate the expected future costs of current decisions, often referred to as the recourse function. In practice, this calculation is computationally difficult as it requires the evaluation of a multidimensional integral whose integrand is an optimization problem. In turn, the recourse function has to be estimated using techniques such as scenario trees or Monte Carlo methods, both of which require numerous functional evaluations to produce accurate results for large-scale problems with multiple periods and high-dimensional uncertainty. In this work, we introduce an importance sampling framework for stochastic programming that can produce accurate estimates of the recourse function using a small number of samples. Previous approaches for importance sampling in stochastic programming were limited to problems where the uncertainty was modeled using discrete random variables, and the recourse function was additively separable in the uncertain dimensions. Our framework avoids these restrictions by pairing Markov Chain Monte Carlo methods with Kernel Density Estimation algorithms to build a non-parametric importance sampling distribution, which can then be used to produce a lower-variance estimate of the recourse function. We demonstrate the increased accuracy and efficiency of our approach using variants of well-known multistage stochastic programming problems. Our numerical results show that our framework produces more accurate estimates of the optimal value of stochastic programming models, especially for problems with moderate variance, multimodal or rare-event distributions.

Key words: Benders' Decomposition, Cutting Plane Algorithms, Stochastic Optimization, Stochastic Programming, Importance Sampling, Variance Reduction, Monte Carlo, Markov Chain Monte Carlo, Kernel Density Estimation, Non-Parametric
History: Last Edited On: November 5, 2013

1. Introduction

Stochastic programming models are large-scale optimization problems that are used to facilitate decision-making under uncertainty. Optimization algorithms for such problems require the evaluation of the expected future costs of current decisions, often referred to as the recourse function. In practice, this calculation is computationally difficult as it requires the evaluation of a multi-dimensional integral whose integrand is an optimization problem. Many algorithms approximate the value of the recourse function using quadrature rules (Pennanen and Koivu (2005)) or Monte Carlo (MC) methods (Shapiro et al. (2009), Birge and Louveaux (2011)). MC methods are particularly appealing for this purpose because they are easy to implement and remain computationally tractable when the recourse function depends on multiple random variables. Nevertheless, the sampling error in MC estimates can significantly impact the results of a stochastic programming model. Although one can reduce the sampling error in MC estimates by using more samples in the MC procedure, this approach is not computationally tractable in stochastic programming because each sample requires the solution to a separate optimization problem. As a result, MC methods need to be paired with a variance reduction technique in order to produce MC estimates with lower sampling error for a moderate number of samples.

In this paper, we focus on a variance reduction technique known as importance sampling. Importance sampling aims to reduce the sampling error of MC estimates by generating samples from an importance sampling distribution. Ideally, this importance sampling distribution is constructed in a manner that favors samples from regions that contribute most to the value of the recourse function. Although many distributions can be used for this purpose, there exists an importance sampling distribution that is optimal in the sense that it can produce MC estimates with zero variance (Asmussen and Glynn (2007)). This so-called zero-variance distribution cannot be used in practice, but it can be used to guide the design of effective importance sampling distributions. Importance sampling was first applied to stochastic programming in a series of papers by Dantzig and Glynn (1990) and Infanger (1992). The importance sampling distribution from these papers showed promising results as it was derived from the zero-variance distribution. Unfortunately, the distribution was developed under the assumptions that the uncertainty is modeled using discrete random variables, and that the cost surface is additively separable in the random dimension - both of which can be restrictive for practical applications.

The primary contribution of this paper is an importance sampling framework that does not require such assumptions. Our framework, which we refer to as the Markov Chain Monte Carlo Importance Sampling (MCMC-IS) framework, exploits the fact that the zero-variance distribution is known up to a normalizing constant. It first uses a Markov Chain Monte Carlo (MCMC) algorithm to generate samples from the zero-variance distribution, and then uses a Kernel Density Estimation (KDE) algorithm to construct an approximate zero-variance distribution from these samples. With this approximate zero-variance distribution at hand, we can then use importance sampling to generate a second set of more relevant samples, and form a lower-variance estimate of the recourse function. MCMC-IS is flexible, in that it accommodates a wide array of MCMC and KDE algorithms; non-parametric, in that it does not require users to specify a family of distributions; robust, in that it readily good results for probability distributions that are difficult to work with using existing methods; and well-suited for stochastic programming, in that it produces lower-variance estimates of the recourse function which ultimately allow us to solve these models more accurately.

Importance sampling is just one of many variance reduction techniques that can be used in stochastic programming. The use of Quasi-Monte Carlo (QMC) methods were studied in Koivu (2005) and in Drew and Homem-de Mello (2006). The non i.i.d. case of MC sampling has been studied in Homem-de Mello (2006). Control variates were proposed in Shapiro and Homem-de Mello (1998) and in Higle (1998). A sequential sampling algorithm was proposed in Bayraksan and Morton

(2011). A computational assessment of conditional sampling, antithetic sampling, control variates and importance sampling appeared in Higle (1998). QMC and Latin Hypercube Sampling (LHS) were compared in Homem-de Mello et al. (2011). The effect of sampling on the solution quality of stochastic programming problems was discussed in Linderoth et al. (2006). In this paper, we use a series of numerical experiments to demonstrate that our proposed framework performs well when compared to Crude Monte Carlo (CMC) methods, Quasi-Monte Carlo (QMC) methods and the importance sampling technique developed in Dantzig and Glynn (1990) and Infanger (1992) (DGI). In addition, we show that our framework significantly outperforms the existing sampling methods when the uncertainty is modeled using a higher variance, rare-event or multi-modal distribution.

MC methods need to be paired with optimization algorithms to solve stochastic programming problems. In this paper, we illustrate the computational performance of MCMC-IS using a popular decomposition algorithm known as the Stochastic Dual Dynamic Programming (SDDP) algorithm Pereira and Pinto (1991). We note, however, that MCMC-IS can be paired with many other stochastic optimization algorithms, such as the sample average approximation method (Shapiro et al. (2009)), stochastic decomposition (Higle and Sen (1991)), progressive hedging (Rockafellar and Wets (1991)), augmented Lagrangian methods (Parpas and Rustem (2007)), variants of Benders' decomposition (Birge and Louveaux (2011)), or even approximate dynamic programming (Powell (2007)). More generally, we also expect MCMC-IS to yield similar benefits in sampling-based approaches for developing stopping rules (Morton (1998)), chance-constrained programming (Watson et al. (2010)), and risk-averse stochastic programming (Shapiro (2009)).

While both MCMC and KDE algorithms have received considerable attention in the literature, they have not - to our knowledge - been used in this way within the realm of stochastic optimization. There has been some recent work on non-parametric importance sampling methods in the field of statistics (Zhang (1996), Neddermeyer (2009)). However, these techniques have not been adopted for practical applications due to the computational overhead involved in building a non-parametric importance distribution. In this paper, we demonstrate that the computational overhead of using an MCMC and KDE procedure in negligible in the context of stochastic programming, and that the MCMC-IS procedure is a highly efficient way to obtain accurate results given a fixed number of functional evaluations or runtime.

Our paper is structured as follows: in Section 2, we provide a brief overview of stochastic programming, and illustrate the mechanism through which decomposition algorithms can produce inaccurate results for a stochastic program when they are paired with a MC method. In Section 3, we introduce the MCMC-IS framework, and present readers with a set of theoretical insights and practical guidelines. In Section 4, we use a series of numerical experiments based on a simple Newsvendor problem to illustrate the sampling-based properties of MCMC-IS, and to demonstrate the benefits of pairing MCMC-IS with a decomposition algorithm to solve stochastic programming models. In Section 5, we then show that these benefits generalize across a collection of benchmark stochastic programming models. We summarize our contributions and discuss directions for future research in Section 6.

2. Motivation

We consider a multistage linear stochastic programming model defined as,

$$\begin{aligned} z^* = \min_{x_1} \quad & c_1^{\mathrm{T}} x_1 + \mathcal{Q}_1(x_1) \\ \text{s.t.} \quad & A_1 x_1 = b_1, \\ & x_1 \geq 0, \end{aligned} \tag{2.1}$$

where $c_1 \in \mathbb{R}^{n_1}$, $A_1 \in \mathbb{R}^{n_1 \times m_1}$ and $b_1 \in \mathbb{R}^{m_1}$. In general, the function $\mathcal{Q}$ is called the recourse function, and is used to represent the expected future costs of current decisions,

$$\mathcal{Q}_t(x_t) = \mathbb{E}[Q_t(x_t, \xi_{t+1})], \quad t = 1, \dots, T-1. \tag{2.2}$$

Given a fixed decision in the previous stage and a realization of the random parameters, the future costs of the model can be estimated by solving the linear program,

$$\begin{aligned} Q_{t-1}(\widehat{x}_{t-1}, \xi_t) = \min_{x_t} \quad & c_t^{\mathrm{T}}(\xi_t) x_t + \mathcal{Q}_t(x_t) \\ \text{s.t.} \quad & A_t(\xi_t) x_t = b_t(\xi_t) - W_t(\xi_t)\widehat{x}_{t-1}, \\ & x_t \geq 0, \end{aligned} \tag{2.3}$$

where $Q_T(\widehat{x}_{T-1}, \xi_T) \equiv 0$ without loss of generality. We will assume that $c_t \in \mathbb{R}^{n_t}$, $A_t \in \mathbb{R}^{n_t \times m_t}$, $W_t \in \mathbb{R}^{n_{t-1} \times m_t}$, $b_t \in \mathbb{R}^{m_t \times 1}$. The components of these parameters are deterministic for $t = 1$, but may be random for $t = 2, \dots, T$. We refer to the set of all random components of the parameters at stage t using a D_t-dimensional random vector ξ_t, and denote its joint probability density function, cumulative distribution function and support as f_t, F_t and Ξ_t respectively. We note that we will frequently drop the time index t from the definition of the recourse function when it is not relevant to the discussion at hand (e.g. when we are referring to two-stage problems). In such cases, we assume that $\mathcal{Q} = \mathcal{Q}_2$, $D = D_2$, $\xi = \xi_2$ and $\hat{x} = \hat{x}_2$. We refer the interested reader to Birge and Louveaux (2011) for an overview of multistage stochastic programming.

Many algorithms have been developed to solve multistage stochastic programming problems. A key step in these algorithms is the discretization of the random parameters. An alternative to discretization is to generate samples of the random parameters, and use an MC method to estimate the recourse function. Such an approach is advantageous in that it can accommodate discrete or continuous random variables, remain computationally tractable for models with a large number of random variables, and produce estimates of the recourse function whose error does not depend on the number of random variables used in the model. Nevertheless, the error of these estimates can significantly alter the results of a stochastic programming model. In what follows, we explain how MC methods can be embedded in decomposition algorithms, and demonstrate how the sampling error of MC estimates can produce inaccurate estimates of the optimal value and solution of a multistage stochastic program.

2.1. The Perils of Sampling in Decomposition Algorithms

Decomposition algorithms are designed to solve multistage stochastic programming problems by constructing a piecewise linear approximation of the epigraph of the recourse function (Birge and Louveaux (2011)). The approximation is composed of supporting hyperplanes to the recourse function at fixed values of x. The supporting hyperplanes are also known as cuts. Given a fixed value $\widehat{x}$, a cut takes the form of a linear inequality constraint,

$$\mathcal{Q}_t(x_t) \geq \mathcal{Q}_t(\widehat{x}_t) + \partial\mathcal{Q}_t(\widehat{x}_t)(x_t - \widehat{x}), \tag{2.4}$$

where $\partial\mathcal{Q}_t$ represents the subgradient of the recourse function. We note that the parameters $\mathcal{Q}_t$ and $\partial\mathcal{Q}_t$ are the expected values of the optimal objective value and dual variables of the linear program in (2.3). The preceding inequality assumes that these parameters can be calculated exactly. This is usually only possible when the random variables in our model have a small, limited number of outcomes. In practice, the expectations are therefore estimated using an MC procedure in which we first generate a set of N samples of the random variables, $\xi_t^1, \dots, \xi_t^N$, and then compute:

$$\widehat{\mathcal{Q}}_t^{MC}(\widehat{x}_t) = \frac{1}{N}\sum_{i=1}^{N} Q_t(\widehat{x}_t, \xi_t^i) \qquad \widehat{\partial\mathcal{Q}}_t^{MC}(\widehat{x}_t) = \frac{1}{N}\sum_{i=1}^{N} \partial Q_t(\widehat{x}_t, \xi_t^i). \tag{2.5}$$

Although MC methods can significantly reduce the computational burden in generating cuts relative to a scenario tree based approach, the cuts generated with MC methods are subject to sampling error. Even if the sampling error associated with each cut is negligible, the errors can compound across the iterations of the decomposition algorithm. As a result, decomposition algorithms that use a small number of samples may produce an invalid approximation of the recourse function which then leads to inaccurate results for the original problem. We illustrate this phenomenon in Figure 1, where we plot the sampled cuts that are produced when a CMC method is paired with a decomposition algorithm in order to solve a simple two-stage Newsvendor problem, whose parameters are specified in Section 4.1.

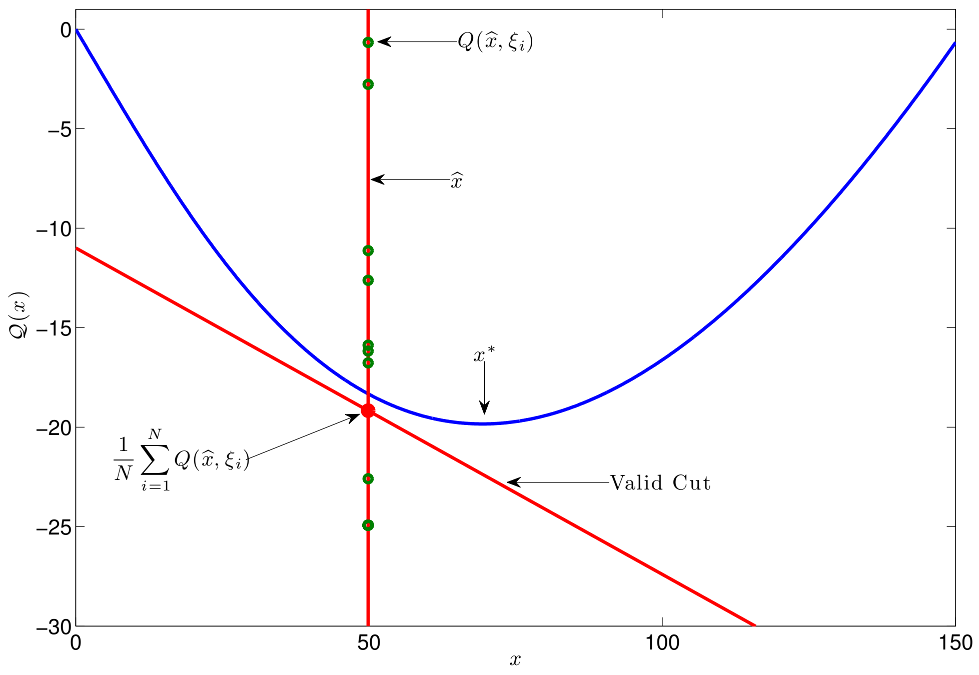
Both cuts in this example were constructed using $N = 50$ samples. For clarity, we plot a subset of the sample values $Q(\widehat{x}, \xi_i), i = 1, \ldots, N$ along the vertical line of $\widehat{x}$, as well as their sample average. In Figure 1(a), we are able to generate a valid sampled cut, which is valid because it underestimates the true recourse function $\mathcal{Q}(x)$ at all values of x. However, it is possible to generate a sampled cut that in some regions overestimates, and in other regions underestimates the true recourse function $\mathcal{Q}(x)$. We illustrate this situation in Figure 1(b), where the sampled cut excludes the true optimal solution at $x^* \approx 69$ with $z^* \approx -20$. Assuming that the algorithm only generates valid cuts until the algorithm converges, the resulting estimates of x^* and z^* will be $\tilde{x} \approx 38$ and $\tilde{z} \approx -15$, corresponding to errors of 80% and 25% respectively.

It is true that we can avoid generating invalid sampled cuts if we model the uncertainty in the problem using a scenario tree. While this approach allows us to calculate the exact values of the parameters in (2.4), it suffers from a different complication. Scenario trees are discrete in nature, and therefore require models where the uncertainty is modeled through discrete random variables, or a suitable discretization procedure that can represent continuous random variables using finite outcomes and probabilities. In the latter case, the scenarios are fixed and the parameters in (2.4) are easy to calculate. However, there are no guarantees that the solution obtained with the discretized scenario tree will be optimal for the original continuous problem unless a large number of scenarios is used. It is an active area of research how to best address this issue and a number of ways have been proposed such as scenario reduction techniques (Dupačová et al. (2003)). Even though scenario trees can yield accurate answers for stochastic programming problems with a small number of random variables and time periods, they still present computational challenges for large-scale problems with many random variables and many time periods. In turn, we focus on the sampled cut approach described previously.

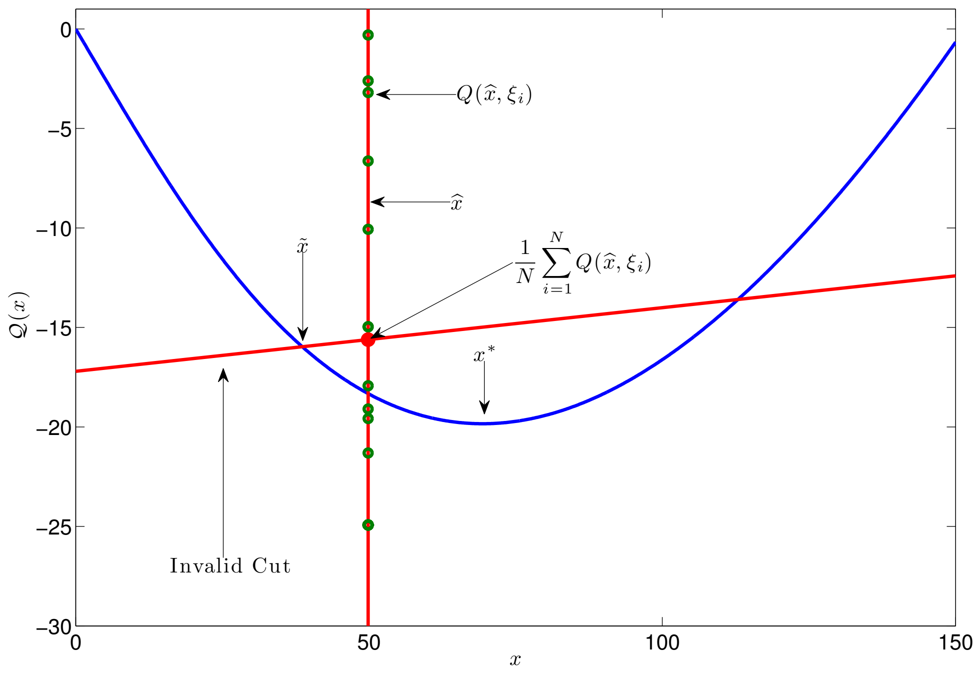
3. The Markov Chain Monte Carlo Approach to Importance Sampling

It is well-known that we can reduce the sampling error in the cut parameters if we increase the number of samples that we use to construct their MC estimates. Even so, the $O(N^{-0.5})$ convergence rate of MC methods effectively implies that we have to solve four times as many linear programs in order to halve the sampling error of the cut parameters. Given the time that is required to solve a typical linear program within a large-scale stochastic programming model, such an approach is simply not tractable. The sampling error of the cut parameters depends on σ^2/N, where σ^2 denotes the variance of the estimate. As a result, an alternative way to reduce the sampling error in the cut parameters without increasing the number of samples is to reduce the underlying variance of the quantity that we are trying to estimate.

Importance sampling is a variance reduction technique that can produce an estimate of $\mathcal{Q}$ which has lower variance and lower sampling error, than $\widehat{\mathcal{Q}}^{MC}$ in (2.5). The variance reduction is achieved by using a different probability distribution that can generate samples in regions that contribute the most to $\mathcal{Q}$. Although importance sampling estimates may have substantially lower-variance than their MC counterparts, choosing a suitable importance sampling distribution is a challenging process that is difficult to generalize and has motivated many papers in the statistics and simulation



(a)



(b)

Figure 1 In 1(a) the sampled cut is valid; assuming that only valid cuts are generated in subsequent iterations, a decomposition algorithm will produce accurate estimates of x^* and z^*. In 1(b) the sampled cut is invalid; even if all the other cuts produced by the algorithm are valid, the true optimal solution at x^* will remain infeasible, and a decomposition algorithm will produce high-error estimates for the optimal value and solution.

literature. We refer the interested reader to Asmussen and Glynn (2007) for a review of importance sampling.

3.1. Importance Sampling and the Curse of Circularity

Importance sampling is a variance reduction technique that constructs lower-variance estimates using an importance sampling distribution g, as opposed to the original sampling distribution f. When samples are generated from the importance sampling distribution g, the recourse function can be calculated as,

$$\mathcal{Q}(\widehat{x}) = \mathbb{E}_f[Q(\widehat{x},\xi)] = \mathbb{E}_g[Q(\widehat{x},\xi)\Lambda(\xi)]. \tag{3.1}$$

where,

$$\Lambda(\xi) = \frac{f(\xi)}{g(\xi)}. \tag{3.2}$$

The function $\Lambda : \Xi \to \mathbb{R}$ is called the likelihood function, and is used to correct the bias introduced by the fact that we generated the samples from g instead of f. In theory, the only requirement for the importance sampling distribution g is that the likelihood function Λ has to be well-defined over the support of f. In other words, $g(\xi) > 0$ at all values of ξ where $f(\xi) > 0$.

Once we select a suitable important sampling distribution g, we can use it to generate a set of N i.i.d. samples $\xi_1 \ldots \xi_N$ and construct an importance sampling estimate of the recourse function as,

$$\widehat{\mathcal{Q}}^{IS}(\widehat{x}) = \frac{1}{N}\sum_{i=1}^{N} Q(\widehat{x},\xi_i)\Lambda(\xi_i). \tag{3.3}$$

The benefit of generating samples from g depends on the amount of variance reduction that can be achieved. Importance sampling is most effective in the context of stochastic programming when g can generate samples from the regions that contribute the most to the value of the recourse function at a fixed point $\widehat{x}$. It is easy to show that the variance of an importance sampling estimate is minimized when we sample from,

$$g^*(\xi) = \frac{|Q(\widehat{x},\xi)|}{\mathbb{E}_f|Q(\widehat{x},\xi)|} f(\xi). \tag{3.4}$$

The importance sampling distribution g^* is optimal in the sense that no other distribution can produce an importance sampling estimate with lower-variance (Asmussen and Glynn (2007)). In fact, if $Q(x,\xi)$ is always positive then g^* produces estimates with zero variance, and is therefore usually referred to as the zero-variance distribution. The problem with using (3.4) in practice is that it requires us to know the value of $\mathbb{E}_f|Q(x,\xi)|$, which is the quantity that we sought to compute in the first place. We are thus faced with a "curse of circularity" in that we can use (3.4) to construct zero-variance estimates if and only if we already have a zero-variance estimate of $\mathbb{E}_f|Q(\widehat{x},\xi)|$.

The importance sampling framework that we introduce in this paper revolves around two key observations. The first observation is that we can generate samples from (3.4) using an MCMC algorithm since we know the distribution up to a normalizing constant $\mathbb{E}^f|Q(x,\xi)|$. This observation is well-known and many MCMC methods have been developed to take advantage of this. We note that we cannot use these samples to form a zero-variance importance sampling estimate because we need to evaluate the likelihood of each sample as shown in (3.2). In this case, the likelihood of a given sample is given by,

$$\Lambda^*(\xi) = \frac{\mathbb{E}_f|Q(x,\xi)|}{|Q(x,\xi)|}, \tag{3.5}$$

and it is also impossible to compute in practice as it depends on $\mathbb{E}_f|Q(x,\xi)|$. This leads us to the second observation: while we cannot use the samples from (3.4) to directly form an importance sampling estimate, we can use them to reconstruct an approximation of the zero-variance distribution using a KDE algorithm. Using this approximate distribution in hand, we then can generate a second set of samples, evaluate the likelihood of each sample, and form a lower-variance importance sampling estimate.

3.2. Description of the MCMC-IS Framework

Our proposed framework consists of three steps: (1) generate samples from the zero-variance distribution using an MCMC algorithm, (2) construct an approximate zero-variance distribution using a KDE algorithm, and (3) sample from the approximate zero-variance distribution to form a lower-variance importance sampling estimate.

MCMC algorithms are an established set of MC methods that can generate samples from a density known up to a normalizing constant. In contrast to other MC methods, MCMC algorithms produce a sequence of serially correlated samples. This sequence forms a Markov Chain whose stationary distribution is the target density, given by (3.4) in our case. Although many different MCMC algorithms can be used within the MCMC-IS framework, we restrict our focus to the Metropolis-Hastings algorithm because it is easy to implement, does not require the specification of many parameters, and does not depend on a restrictive set of assumptions. We refer the interested reader to Gelman et al. (2010) for more on the Metropolis-Hastings algorithm, and other MCMC algorithms that can be used in MCMC-IS.

The Metropolis-Hastings algorithm uses a simple accept-reject procedure in order to generate a Markov Chain that has (3.4) as its stationary distribution. In the k-th step, the algorithm generates a proposed state ζ_k using a proposal distribution whose density $q(\cdot \mid \xi_k)$ typically depends on the current state ξ_k. Together, the proposed state, the current state and the target density are used to evaluate an acceptance probability, $a(\xi_k, \zeta_k)$. The proposed state is accepted with probability $a(\xi_k, \zeta_k)$, in which case the Markov Chain transitions to the proposed state $\xi_{k+1} := \zeta_k$. Otherwise, the proposed state is rejected with probability $1 - a(\xi_k, \zeta_k)$, in which case the Markov Chain remains at its current state $\xi_{k+1} := \xi_k$.

In this paper, we use a special instance of the Metropolis-Hastings algorithm in which new states are proposed using a random walk process. This implies that the proposed state ζ_k at each step of the Metropolis-Hastings algorithm is generated as,

$$\zeta_k = \xi_k + v_k, \tag{3.6}$$

where v_k is a D-dimensional Gaussian random variable with mean 0 and covariance matrix Σ. In practice, the Metropolis-Hastings algorithm requires that Σ is specified beforehand. However, we can avoid specifying this parameter if we use the Adaptive Metropolis algorithm described in Haario et al. (2001). When states are proposed through a random walk process, the proposal distribution is symmetric and the acceptance probability can be expressed as,

$$a(\xi_k, \zeta_k) = \min\left\{1, \frac{|Q(\widehat{x}, \zeta_k)| f(\zeta_k)}{|Q(\widehat{x}, \xi_k)| f(\xi_k)}\right\}. \tag{3.7}$$

Once a set of M samples has been generated from the zero-variance distribution specified in (3.4) using an MCMC algorithm, we can construct an approximate zero-variance distribution from these samples using a Kernel Density Estimation (KDE) algorithm. KDE algorithms are established techniques that are used to reconstruct continuous probability distributions from a finite set of samples. We refer the interested reader to Devroye and Györfi (1985), Silverman (1986) and Scott (1992) for a detailed overview of these techniques. It is worth noting that a KDE algorithm can construct the approximate zero-variance distribution, even as the MCMC algorithm produces correlated samples (Hall et al. (1995))

The probability density function generated by the KDE methodology is given by,

$$\widehat{g}_M(\xi) = \frac{1}{M}\sum_{i=1}^{M} K_H(\xi, \xi_i), \tag{3.8}$$

where the function K_H is referred to as a kernel function, and $H \in \mathbb{R}^{D\times D}$ is its associated bandwidth matrix. In order to ensure that $\widehat{g}$ is a proper probability density function, we impose the following conditions on the kernel function,

$$\begin{aligned} K_H(\cdot,\cdot) &\geq 0, \\ \int_\Xi K_H(\xi,\cdot)d\xi &= 1. \end{aligned} \tag{3.9}$$

In addition, we assume that the kernel matrix is positive semidefinite, meaning that the matrix with $(i,j)^{th}$ entry given by $K_H(\xi_i,\xi_j),\ 1 \leq i,j \leq M$ is positive semidefinite. These assumptions are required by most KDE algorithms, and are satisfied by the majority of kernels used in practice. A popular kernel, and the one that we use in this paper, is the Gaussian product kernel,

$$K_H(\xi,\xi_i) = \prod_{k=1}^{D} \frac{1}{\sqrt{2\pi}h_k} \exp\left(\frac{(\xi_k - \xi_{i,k})^2}{2h_k^2}\right). \tag{3.10}$$

The associated bandwidth matrix H for the Gaussian product kernel is a $D \times D$ diagonal matrix that contains the bandwidth parameters of each dimension $h_1,\ldots,h_D$ along its diagonal. In our implementation, we use a one-dimensional likelihood-based search to estimate the value of the bandwidth parameter h_k separately for each dimension k.

Using the approximate zero-variance distribution $\widehat{g}_M$, we can finally construct an importance sampling estimate of the recourse function by generating N additional samples from $\widehat{g}_M$. Although these samples will not originate from the true zero-variance distribution g^*, they can still be used to produce a lower-variance importance sampling estimate provided that the KDE algorithm has constructed a $\widehat{g}_M$ that is similar to g^*. Generating samples from $\widehat{g}_M$ is also beneficial in that the samples are independent and the kernel functions are easy to sample from. In practice, we construct $\widehat{g}_M$ using modest values of M and then construct $\widehat{\mathcal{Q}}^{IS}(\widehat{x})$ using large values of N.

The computational burden of the MCMC step is a result of the accept-reject algorithm which typically requires more LP evaluations (proposed samples) than are used (accepted samples). The additional advantage of estimating and sampling the approximate importance sampling distribution is the relative efficiency of generating a larger number of samples.

We provide a full formal description of the MCMC-IS framework in Algorithm 1.

3.3. Ingredients of the Convergence Analysis for MCMC-IS

MCMC-IS has two sources of error. The first source of error is due to the MCMC algorithm used to generate samples from the zero-variance distribution, and the second is due to the KDE algorithm used in the construction of the approximate zero-variance distribution. If the sampling algorithm is embedded within an optimization algorithm then there is also a third source of error, but in this section we focus on the sampling aspect. The main convergence condition for an MCMC algorithm requires the underlying Markov chain to be irreducible and aperiodic. The irreducibility property means that the chain can eventually reach any subset of the space from any state. The aperiodic condition means that the chain cannot return to a subset of the space in a predictable manner. Formal definitions of these properties can be found in Roberts and Rosenthal (2004). The first step in the convergence analysis is to show that these two conditions are satisfied. In the case of the SDDP algorithm, the MCMC algorithm will be used whenever a new sampled cut needs to be

Algorithm 1 Markov Chain Monte Carlo Importance Sampling (MCMC–IS)

```
Require: $\widehat{x}$: previous stage decision
Require: $M$: number of samples generated using the MCMC algorithm
Require: $N$: number of samples generated using the approximate zero-variance distribution
Require: $\xi_0$: starting state of the MCMC algorithm

    Step 1: Generate Samples from the Zero-Variance Distribution using MCMC

1: Set $k = 0$
2: Given the current state $\xi_k$, generate $\zeta_k \sim q(\cdot \mid \xi_k)$.
3: Generate a uniform random variable $u \sim U \in (0,1)$.
4: Transition to the next state according to,
       $$\xi_{k+1} = \begin{cases} \zeta_k & \text{if } \; u \leq a(\xi_k, \zeta_k) \\ \xi_k & \text{otherwise} \end{cases},$$
   where,
       $$a(\xi_k, \zeta_k) = \min\left\{1, \frac{|Q(\widehat{x}, \zeta_k)| f(\zeta_k) q(\xi_k|\zeta_k)}{|Q(\widehat{x}, \xi_k)| f(\xi_k) q(\zeta_k|\xi_k)}\right\}$$
5: Let $k \leftarrow k+1$. If $k = M$ then proceed to Step 6. Otherwise return to Step 2.

    Step 2: Construct the Zero-Variance Distribution using KDE

6: For each state of the Markov chain generated from MCMC, reconstruct the approximate zero-
   variance distribution as,
       $$\widehat{g}_M(\xi) = \frac{1}{M} \sum_{i=1}^{M} K_H(\xi, \xi_i).$$

    Step 3: Sample from the Approximate Zero-Variance Distribution to Form an
    Importance Sampling Estimate

7: Generate $N$ new samples from $\widehat{g}_M$ and form the importance sampling estimate,
       $$\widehat{Q}^{IS}(\widehat{x}) = \frac{1}{N} \sum_{i=1}^{N} Q(\widehat{x}, \xi_i) \frac{f(\xi_i)}{\widehat{g}_M(\xi_i)}$$
```

generated and therefore these two conditions will hold even if the problem does not have complete recourse.

In order to control the error due to the KDE algorithm, we need to ensure that the number of samples generated by the MCMC algorithm M is large enough, and that the bandwidth parameter h_k is small enough. In particular, if $(Mh^D)^{-1} \to \infty$, $h \to 0$ as $M \to \infty$, and the density function is approximated as,

$$\widehat{g}_M(\xi) = \frac{1}{M} \sum_{i=1}^{M} K_H(\xi, \xi_i) = (Mh^D)^{-1} \sum_{i=1}^{M} K\left(\frac{\xi - \xi_i}{h}\right),$$

then it has been shown that $\widehat{g}_M$ will probabilistically converge to g^* under the total variation norm (see Devroye and Györfi (1985)). Applying this result to the MCMC-IS framework is not straightforward. The complexity stems from the fact that the previous convergence proofs for the KDE algorithm assume that samples are generated from g^*, whereas in our framework these samples are generated from a Markov chain whose stationary distribution is g^*.

3.4. Practical Guidelines for MCMC-IS

The first important choice for MCMC-IS is the choice of a proper MCMC algorithm and a suitable proposal distribution. In our experiments, we have used our own implementation of the Metropolis-Hastings MCMC algorithm and the Adaptive Metropolis MCMC algorithm described in Haario et al. (2001). Both algorithms propose new samples using a random walk process that starts off at a user-defined point ξ_0, which we set as $\xi_0 = \mathbb{E}_f[\xi]$. In turn, the main benefit of the Adaptive Metropolis algorithm is that it does not require users to specify the step-size for the random walk process. More specifically, the Adaptive Metropolis algorithm uses a random walk process in which the steps are normally distributed with zero mean and the identity matrix as the covariance matrix - all the while keeping track of accepted samples. After a fixed number of iterations (in our case, 30 per dimension of the random vector ξ), the Adaptive Metropolis algorithm begins to use a sample covariance matrix that is estimated from previously accepted samples.

Another important choice in implementing MCMC-IS is the number of samples to generate using an MCMC algorithm (M). This is an important choice since generating samples with a MCMC algorithm is computationally expensive due to the fact that it often takes more than M functional evaluations to obtain M samples (as some samples are rejected in the MCMC process). In our experience, we have found that a small number of samples produces a significant amount of variance reduction. Accordingly, we have used $M = 3000$ within all of our numerical experiments in Sections 4 and 5. It may be surprising that a small and constant number is sufficient even for large-scale problems. In Section 4.3 we provide a possible explanation for this result based on our numerical experiments. In particular, it seems that a small number of samples is sufficient to bias the sampling towards the right direction, and that the computational advantage of sampling from the "exact" density is relatively small compared to the computational cost of computing it.

The final choice in implementing MCMC-IS related to the KDE algorithm that is used to construct the approximation zero-variance distribution. In our experiments, we have used the MATLAB KDE Toolbox, which is available online at `http://www.ics.uci.edu/~ihler/code/kde.html`. The MATLAB KDE Toolbox is a fast and flexible KDE implementation which allows users to reconstruct kernel density estimates using different types of kernels (e.g. Gaussian, Laplacian and Epatchenikov kernels) as well as different types of bandwidth estimation procedures (e.g. leave-one-out cross-validation, optimizing MISE and AMISE criteria). In our case, we have reconstructed the approximated density using a simple Gaussian product kernel a leave-out-out cross-validation bandwidth estimator. Our experience to date has shown that MCMC-IS is robust with regards to the choice of kernel function using a decent number of samples to reconstruct the approximation zero-variance distribution. Insights into the choice of the bandwidth estimator are provided in Section 4.3.

4. Numerical Experiments with the Newsvendor Problem

In this section, we demonstrate several properties of MCMC-IS using a series of numerical experiments based on a simple two-stage Newsvendor problem. In Sections 4.3 - 4.5, we illustrate different sampling-related properties of MCMC-IS to provide insights into how MCMC-IS works and how it should be used in practice. In Section 4.6, we compare the performance of MCMC-IS estimates to estimates that are produced using a Crude Monte Carlo method (CMC), a Quasi Monte Carlo (QMC) method, and the Dantzig-Glynn-Infanger (DGI) importance sampling technique proposed in Dantzig and Glynn (1990) and Infanger (1992). The remaining numerical experiments in Section 4.8 focus on the performance of MCMC-IS when it is embedded in a decomposition algorithm and used to solve stochastic programming models with different types of uncertainty.

4.1. Description of the Newsvendor Problem

We consider a two-stage Newsvendor problem with uncertain demand and uncertain sales prices, where the first-stage decision-making problem is a linear program defined as,

$$\begin{aligned} z^* = \min_{x} \quad & x + \mathcal{Q}(x_1) \\ \text{s.t.} \quad & x \geq 0, \end{aligned} \tag{4.1}$$

and the recourse function is a linear program defined as,

$$\begin{aligned} Q(\widehat{x}, \xi) = \min_{y_1, y_2} \quad & -p(\xi) y_1 - r y_2 \\ & y_1 \leq d(\xi), \\ & y_1 + y_2 \leq \widehat{x}, \\ & y_1,\ y_2 \geq 0, \end{aligned} \tag{4.2}$$

where $\widehat{x}$ denotes the quantity of newspapers purchased in the first stage, $\xi = (\xi_1, \xi_2)$ represents the uncertainty in demand $d(\xi)$ and sales price $p(\xi)$ of newspapers in the second-stage, and scalar r represents the price of recycling unsold newspapers. We usually model the uncertainty in demand as $d(\xi) = 100 \times \exp(\xi_1)$ and the uncertainty in sales price as $p(\xi) = 1.5 \times \exp(\xi_2)$, where ξ_1 and ξ_2 are normal random variables with mean μ and and standard deviation σ. This implies that the uncertainty in $d(\xi)$ and $q(\xi)$ are modeled using a lognormal distribution. We set $\mu = 0$ and change the underlying variance of the model by altering the value of σ from $\sigma = 1$ to $\sigma = 2$.

4.2. Details on Experimental Setup and Reported Results

Our choice of a simple model for this section is due to the fact that the distributions can be easily visualized, and we can determine the value of the true recourse function at various points using numerical integration procedures. In contrast to other test problems in the stochastic programming literature, this setup allows us to calculate the true values of the optimal solution x^* and the optimal value z^* of the underlying model. In turn, we are able to report the following set of statistics:

- Mean-squared error of the estimate of optimal solution $\tilde{x}$, defined as $\text{MSE}(\tilde{x}) \equiv \|x^* - \tilde{x}\|_2$;
- Mean-squared error of the estimate of the optimal value $\tilde{z}$, defined as $\text{MSE}(\tilde{z}) \equiv \|z^* - \tilde{z}\|_2$;
- Mean-squared error of the approximate zero-variance distribution, defined as $\text{MSE}(\widehat{g}) \equiv \int (g(\xi) - \widehat{g}(\xi))^2 d\xi$;
- Mean-squared error of the estimated recourse function $\widehat{\mathcal{Q}}$ at a fixed point $\widehat{x}$, defined as $\text{MSE}(\widehat{\mathcal{Q}}(\widehat{x})) \equiv \|\mathcal{Q}(x) - \widehat{\mathcal{Q}}(\widehat{x})\|_2$;
- Sample variance of the estimated recourse function $\widehat{\mathcal{Q}}$ at a fixed point $\widehat{x}$, defined as $\text{S}(\widehat{\mathcal{Q}}(\widehat{x}))^2 \equiv \mathbb{E}\big[\mathcal{Q}(x) - \mathbb{E}[\widehat{\mathcal{Q}}(\widehat{x})]\big]^2$;

Such statistics are crucial in measuring the effectiveness of importance sampling procedures as importance sampling estimates will typically have low sample variance, but may be prone to high bias and high mean-squared error. In our experiments, we compute sample average values for these statistics using a total of 30 simulations. We note that have normalized all of these values for the sake of clarity.

Our numerical experiments were specifically designed to provide a fair computational comparison between different sampling methods by ensuring that each sampling method was allotted the same number of functional evaluations. The careful reader should notice that an MCMC-IS uses a total of $M + M_r$ total functional evaluations to construct an importance sampling distribution, where M_r denotes the number of samples that are rejected due to the accept-reject procedure of the MCMC algorithm. Thus, if we ran an instance of MCMC-IS using M samples to construct the importance sampling distribution and N samples to compute our estimate, then we formed a comparable estimate for CMC, QMC and DGI using a total of $M + M_r + N$ samples. We note that this point may be neglected as we have consistently used N to denote the number of samples in Figures and Tables for the sake of clarity. In this case, N only refers to the number of samples used to construct MCMC-IS estimates, and we stress that all other methods were given the same number of functional evaluations as MCMC-IS.

All of the results from our numerical experiments were produced using MATLAB 2012a. In particular, we used a Mersenne-Twister algorithm to generate random numbers that were used for CMC sampling as well as importance sampling procedures. For QMC sampling, we used a Sobol sequence that was randomized using the Matousek-Affine-Owen scrambling algorithm. We note that we have implemented our own version of the DGI importance sampling method, as it is described in Infanger (1992). As stated in Section 3.4, we used our own implementations of the Metropolis-Hastings MCMC algorithm and the Adaptive Metropolis algorithm as the MCMC algorithm in MCMC-IS. In both cases, we produced an approximate zero-variance distribution using the Gaussian product kernel function and a leave-one-out cross-validation bandwidth estimator from the MATLAB KDE Toolbox.

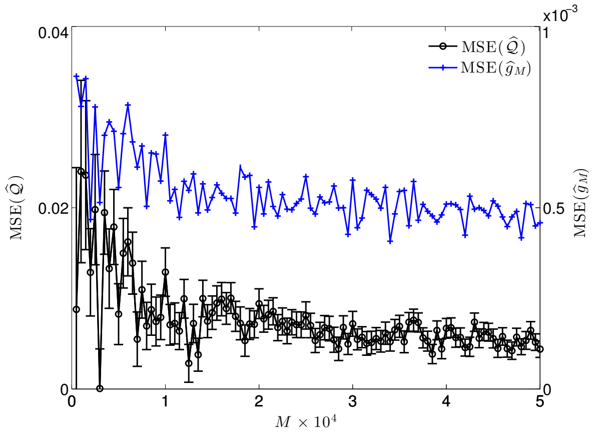
Lastly, all of our stochastic programming problems were solved using a MATLAB implementation of the SDDP algorithm, which used a MEX file to call the IBM ILOG CPLEX 12.4 Callable Library and solve a series of linear programs with sampled parameters in C. We have this set of MEX files to make it easier for practitioners to implement MCMC-IS using MATLAB and CPLEX 12.4 at the first authors' website - `http://www.doc.ic.ac.uk/~pp500/`. The collection includes: a MEX file that can generate a sampled cut; a MEX file that can generate samples from the zero-variance distribution using a Metroplis-Hastings algorithm; and a MEX file that can generate samples from the the zero-variance distribution using an Adaptive Metropolis algorithm. These files can be paired with the MATLAB KDE toolbox and embedded in a decomposition algorithm in order to solve stochastic programming models using MCMC-IS.

4.3. The Effect of the Number of MCMC Samples and the KDE Bandwidth Parameter

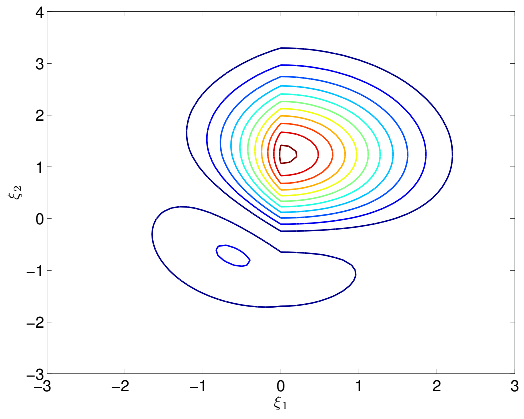
Our numerical experiments with the Newsvendor problem suggest that a modest number of MCMC samples (M) can produce an approximate zero-variance distribution ($\widehat{g}_M$) that yields substantial variance reduction in our estimates of the recourse function.

As shown in Figure 2(a), increasing M does reduce the error in our $\widehat{g}_M$. However, the computational cost of such an increase is not justified in terms of the marginal improvement in the accuracy of our recourse function estimates. This is a positive result as the MCMC algorithm represents a computationally expensive part of our framework. A possible explanation for this empirical observation is that if our $\widehat{g}_M$ qualitatively agrees with g^*, then the sample statistics of the approximate distribution will qualitatively agree with the sample statistics of the zero-variance density.

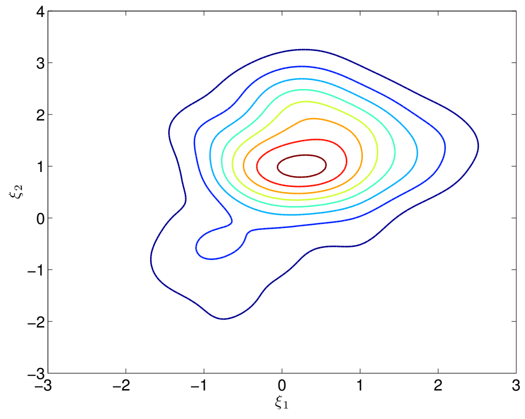
In order to illustrate this point, we plot the contours of the true zero-variance distribution g^* in Figure 2(b) and the contours of $\widehat{g}_M$ for different values of M in Figure 2(c)-2(e). These figures suggest that the approximate distributions produced by MCMC-IS qualitatively agree with the true zero-variance distribution even at low values of M. In Figure 2(f), we show the contours of our approximate zero-variance distribution after we reduce the bandwidth parameters of the kernel function by 20%. This decreases the MSE of $\widehat{g}_M$ by approximately 12% but increases its variance by approximately 15%, thereby demonstrating the bias-variance tradeoff of KDE algorithms.



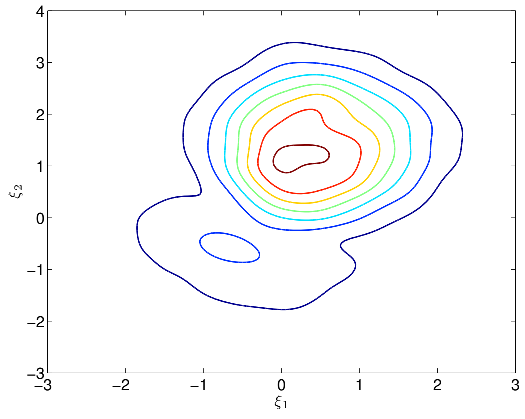
(a)



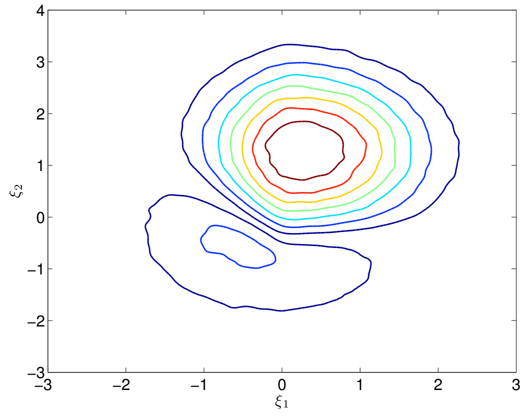
(b) g^*



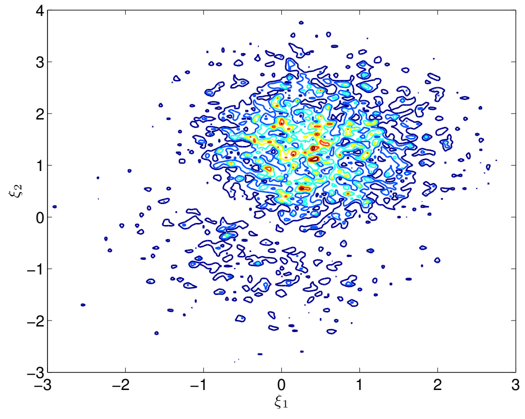
(c) $\widehat{g}_{500}$



(d) $\widehat{g}_{1000}$



(e) $\widehat{g}_{10000}$



(f) $\widehat{g}_{10000}$

Figure 2 (a) The majority of the gains in variance reduction and accuracy can be achieved for a small values of M. Note that the axis for MSE($\widehat{g}_M$) is the right, and the scale for MSE($\widehat{Q}$)is on the left. (b) Contours of g^*. (c)-(e) Contours of $\widehat{g}_M$ for different values of M; the bandwidth parameter for these distributions is estimated using a one-dimensional likelihood-based search. (f) $\widehat{g}_{10000}$ with a bandwidth that is 20% smaller for each dimension. The resulting mean square error is lower but the variance is higher for the density in (f)

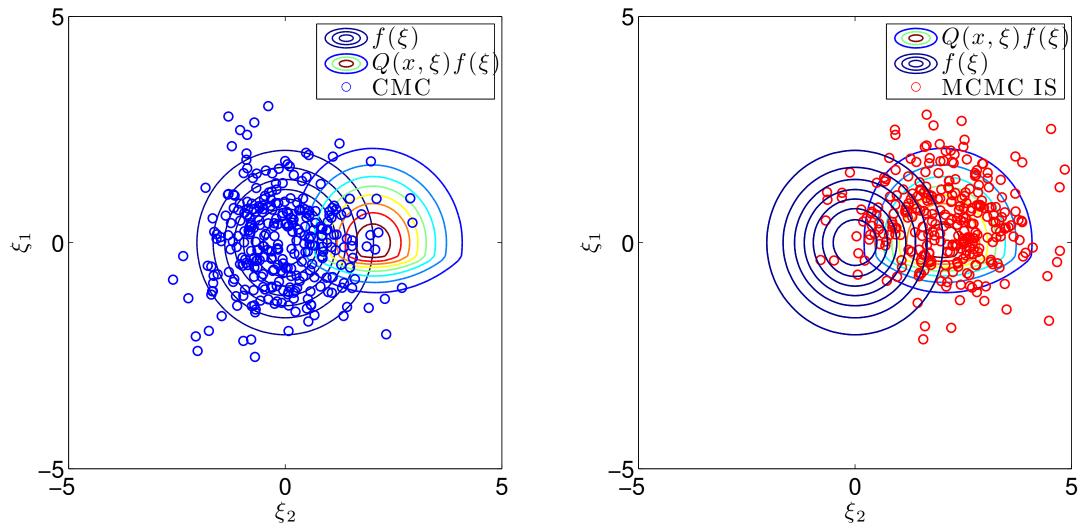


Figure 3 Comparison of points generated with the standard CMC approach, and MCMC-IS. Left: using MC sampling; Right: using importance sampling.

4.4. Adaptive Sampling of the Important Regions

The major difference between our framework and a standard MC method is that we generate samples using an importance sampling distribution $\widehat{g}_M$ as opposed to the original distribution f. As a result, the samples that are generated using $\widehat{g}_M$ are typically located in regions that contribute the most to the value of the recourse function (i.e., in regions where $|Q(\hat{x},\xi)|f(\xi)$ is large) while the samples that are generated using f are typically located in regions where the original distribution attains high values. We demonstrate this difference in Figure 3 where we plot a set of samples generated from f using the CMC method (left), and another set of samples generated from $\widehat{g}_M$ using MCMC-IS. The first set of contours in Figure 3 pertains to the original distribution f while the second set of contours pertains to the true zero-variance distribution g^*. Note that f and g^* are not only centered around different points but also have different shapes. These results were constructed with the first stage decision fixed at $\hat{x}=50$.

4.5. Dependence of the Sampling Distribution on the Previous Stage Decision

In many cases, the importance sampling distributions used within a stochastic programming application should depend on the previous stage decision. We illustrate such a dependence in Figure 4, where we plot the absolute difference between an approximate zero-variance distribution constructed around the point $\widehat{x}_r = 50$ and two other approximate zero-variance distributions constructed around the point $\widehat{x}_1 = 10$ (left) and $\widehat{x}_2 = 100$ (right). As shown, the approximate zero-variance distribution produced by MCMC-IS can vary substantially as we change the previous stage decision.

4.6. Comparison with Other Sampling Algorithms

In this section, we compare MCMC-IS estimates to those produced by the CMC, QMC and DGI methods. In Figure 5(b), we plot the sample standard deviation of the different methods. Although both importance sampling methods perform well in this respect, it is worth noting that MCMC-IS performs better for smaller sample sizes. When we plot the error in Figure 5(a), we find that MCMC-IS and the QMC sampling method perform best.

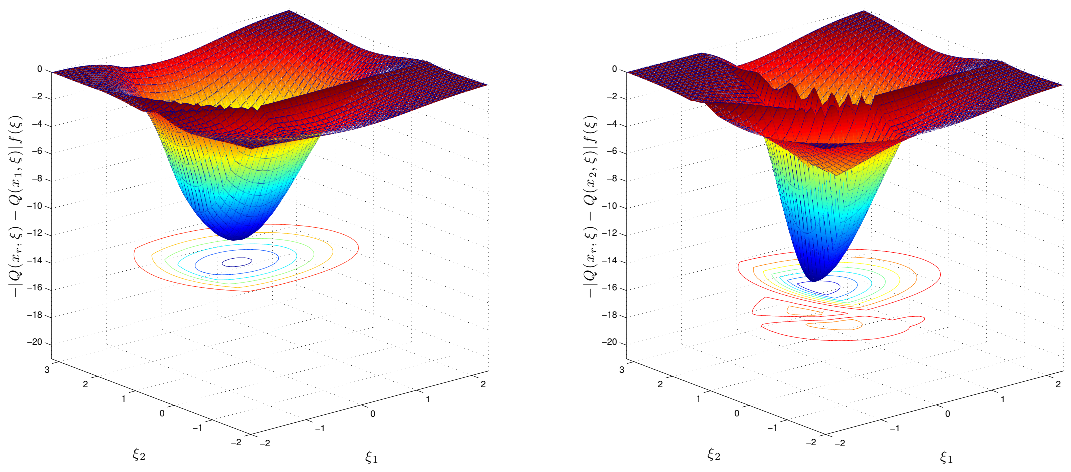


Figure 4 The absolute difference between an approximate zero-variance distribution constructed at $\widehat{x}_r = 50$ and two other approximate zero-variance distributions constructed at $\widehat{x}_1 = 10$ (left) and $\widehat{x}_2 = 100$ (right).

Our results suggest that the relative advantage of MCMC-IS over other variance reduction methods becomes more significant as there is more uncertainty in our model. Increasing the variance of the underlying model typically means that more samples are required for the algorithms to produce estimates with a comparable variance and error. This is to be expected since the error of an MC estimate depends on the variance of the random parameters as well as the sample size. To emphasize this point, we repeat the same calculations as above but increase the standard deviation of (ξ_1, ξ_2) from $\sigma = 1$ to $\sigma = 2$ as described in Section 4.1. In this regime, MCMC-IS outperforms all other methods (Figure 5(c) and 5(d)).

We note that the error in the DGI estimates of the recourse function converges very slowly in this example because the DGI method uses an approximate zero-variance distribution which is specifically built for a recourse function that is additively separable in the random variables. For this problem, however, the recourse function is not additively separable. This leads to estimates of the recourse function that have high variance, and high MSE.

4.7. Multimodal Distributions and Rare-Event Simulation

Many decision-making models involve probability distributions that are multimodal (Bucklew (2004)) or that involve rare-events (Ravindran (2008)). Unfortunately, such complex probability distributions are difficult to include in stochastic programming models as existing variance reduction methods will need an extremely large number of samples in order to generate accurate and reliable results.

Even as importance sampling is frequently used when dealing with such models, existing importance sampling techniques are ill-suited for this purpose due to two reasons. First, as was illustrated in the previous section, an ideal importance sampling distribution depends on the incumbent solution and has to be created each time we wish to generate a new sampled cut. This implies that efficiency is an important consideration. Second, stochastic programming models not only require us to generate samples from these complex distributions, but to use them to compute an accurate estimate of the recourse function. In other words, an appropriate importance sampling technique also must be able to accurately evaluate the likelihood of each sample that it generates as in (3.2) or risk generating biased results. Such issues often preclude the application of stochastic programming when the distribution of the uncertain variables has a complex structure.

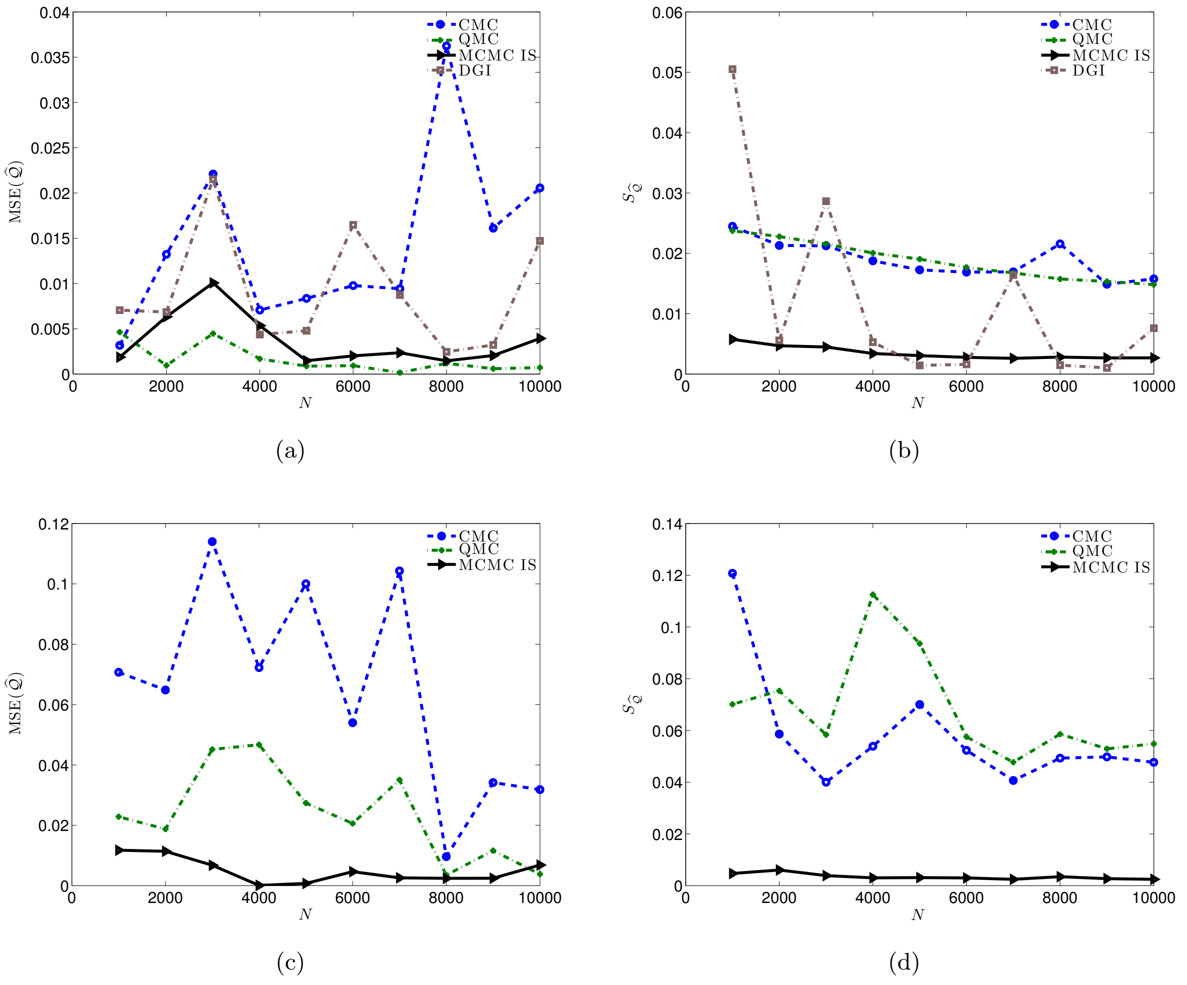
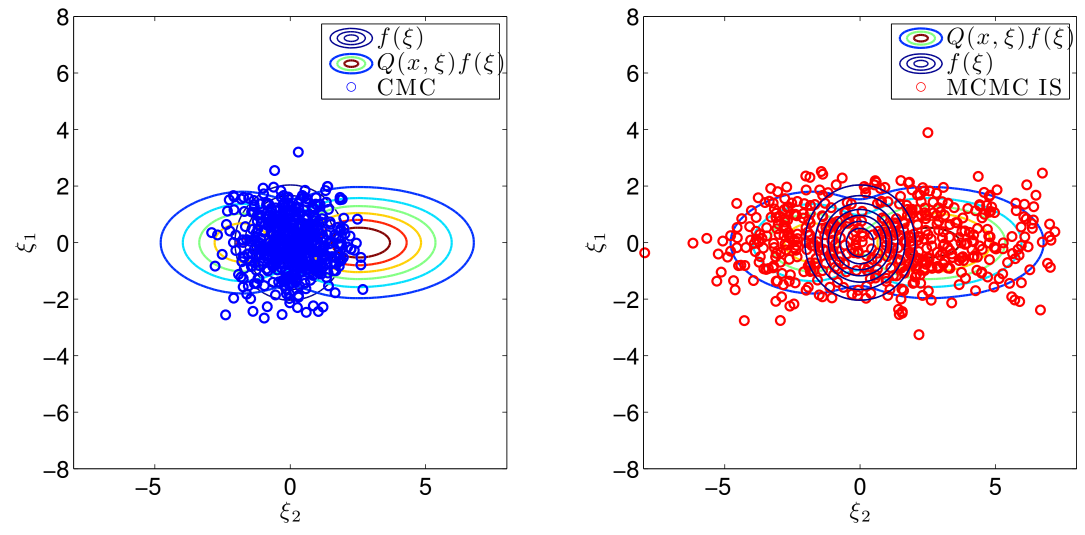


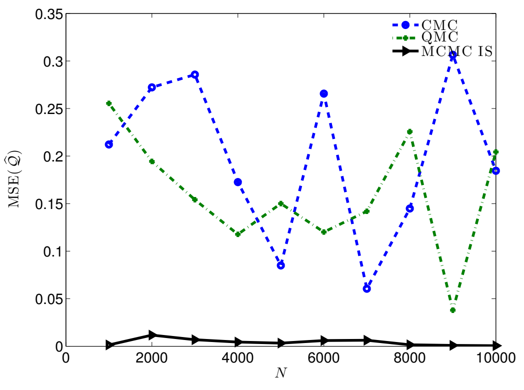
Figure 5 **Top:** Comparison of the accuracy and variance of estimates produced by different methods for a moderate-variance problem with $\sigma = 1$. **Bottom:** Comparison of the accuracy and variance of estimates produced by different methods for a higher-variance problem with $\sigma = 2$. Note that we omit the results for the IDG method when $\sigma = 2$ for clarity. The normalized values of $S_{\widehat{Q}}$ and $\mathrm{MSE}(\widehat{Q})$ for DGI are around 20% and 40% respectively

To demonstrate these issues and show that our proposed algorithm can sample efficiently in such cases, we use an example where the important regions of the recourse function are described by a surface with two distinct modes, whose contours are shown in 6(a). In this example, we have replaced the original integrand in the recourse function $Q(\widehat{x}, \xi_1, \xi_2) f(\xi_1, \xi_2)$ with a new integrand $Q(\widehat{x}, w(\xi_1), w(\xi_2)) f(\xi_1, \xi_2)$, in which $w(\xi) = \exp(\frac{\xi^2}{2} - \frac{(\xi+3)^2}{8}) + \exp(\frac{\xi^2}{2} - \frac{(\xi+1)^2}{8})$, $\widehat{x} = 50$ and f denotes the standard bivariate normal density. This example illustrates rare-event sampling, in the sense that the majority of the samples from the important regions are outside of the 2σ interval of the original distribution, f.

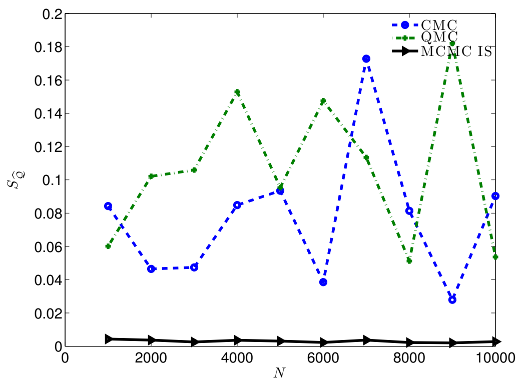
As in Section 4.5, we then generate a set of samples using the CMC method and MCMC-IS. In this example, the samples that are generated using the CMC method are centered around the origin, where the original distribution f attains its highest values (Figure 6(a), left). In contrast, the samples that are generated using MCMC-IS are centered around the two modes and in proportion to the depth of each mode. These areas constitute the regions that contribute the most to the value of the recourse function and correspond to the areas where the approximate zero-variance



(a)



(b)



(c)

Figure 6 **Top**: Contours of a multimodal model. Samples generated using CMC are shown on the left and the samples from MCMC-IS are shown on the right. **Bottom:** Error and variance of estimates produced by different methods.

distribution $\widehat{g}_M$ takes on its largest values. As a result the MCMC-IS framework obtains an estimate of the recourse function that is both more accurate (Figure 6(b)) and has less variance (Figure 6(c)) than the other methods. In this example, we have omitted the results for the DGI method because the importance sampling weights turn out to be zero for all the samples, meaning that the estimates it produces do not converge. This is a well-known problem with the DGI method that has previously been discussed in Section 1.4 of Higle (1998).

4.8. Accuracy and Variance of MCMC-IS Estimates from a Decomposition Algorithm

In this section, we compare the estimates of the optimal value $\tilde{z}$ of the Newsvendor problem when it is solved with a decomposition algorithm which has been paired with MCMC-IS, CMC and QMC.

We consider an extension of the Newsvendor problem from Section 4.1, where the Newsvendor buys and sells s different types of newspapers. We purposely do not include any constraints to couple the different types of newspapers so that we can extrapolate the true values of x^* and z^* for the extended problem using the true values from Section 4.1. In this case, we can assess the accuracy of our estimates for a $D = 2 \times s$ dimensional problem by noting that the optimal solution x^* has to be the same for each of the s different types of newspapers, and the optimal value z^* has to scale additively with the number of different newspapers s.

In contrast to the experiments in Sections 4.3 to 4.7, the accuracy of $\tilde{z}$ depends on the number of sampled cuts that are added to the first-stage problem through a decomposition algorithm, as well as the sampling method that is used to generate these estimates. Note that in our implementation of SDDP, we consider the number of iterations as equivalent to the number of cuts added to the first stage problem. In practice, the number of iterations needed for the algorithm to converge is determined by a stopping test that is designed to assess whether the decomposition algorithm has converged. In this experiment, however, we compare estimates that are produced after a fixed number of iterations. Fixing the number of iterations ensures that each sampling method produces estimates using the same number of samples, and isolates the performance of the sampling method from the performance of the stopping test. During our numerical experiments we fixed the number of iterations to $8 \times s$. We found that this simple rule was sufficient to show the numerical properties of the different sampling algorithms.

Figure 7 shows the convergence of the estimates that we obtain when we solve a two-stage Newsvendor problem with $D = 2 \times 3 = 6$ random variables after $8 \times 3 = 24$ cuts have been added to the first-stage problem. In Figures 7(a) - 7(d), we show the results when we model the uncertainty in the demand and sales price of each newspaper using the lognormal distributions from Section 4.1, and we build the approximate zero-variance distribution for each sampled cut using $M = 3000$ samples that are generated from a standard Metropolis Hastings MCMC algorithm. In Figures 7(e) - 7(f), we show results when we model the uncertainty in the demand and sales price of each newspaper using the multimodal rare-event distribution from Section 4.7, and we build the approximate zero-variance distribution for each sampled cut using $M = 3000$ samples that are generated from the Adaptive Metropolis algorithm described in Haario et al. (2001).

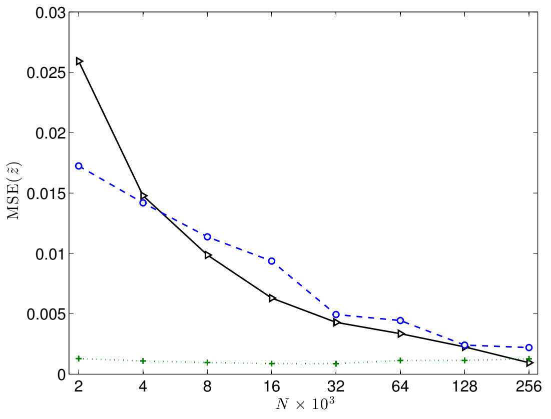
Our results confirm that the relative advantage of using MCMC-IS estimates depends on the inherent variance of the underlying stochastic programming model. In models where the uncertainty is modeled using a lower-variance distribution, MCMC-IS produces estimates that are just as accurate as the estimates produced by a QMC method, but that are still more accurate than the estimates produced by a CMC method. In models where the uncertainty is modeled using a higher-variance or rare-event distribution, MCMC-IS produces estimates that are much more accurate than those produced by QMC and CMC methods. Our numerical results also suggest that MCMC-IS produces estimates with sample standard deviations that are far lower than the estimates produced by CMC and QMC methods.

5. Numerical Experiments on a Collection of Test Problems

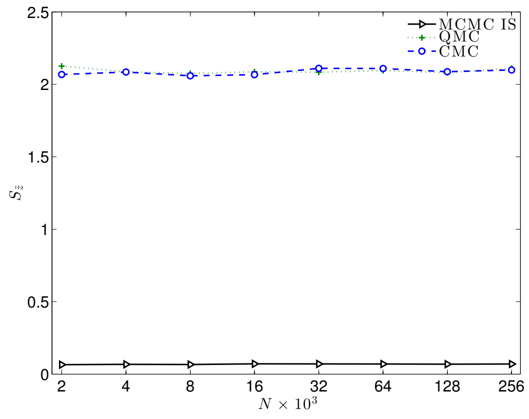
In this section, we demonstrate the performance of MCMC-IS when it is paired with a decomposition algorithm in order to to solve a collection of benchmark stochastic programming models.

5.1. Overview of the Test Problems

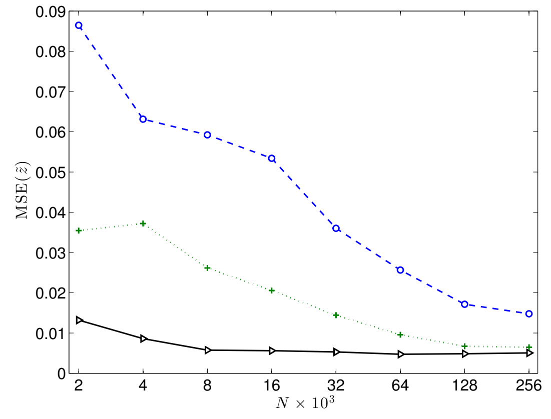
In order to verify that our findings from Section 4.8 generalize to stochastic programming models, we have based the numerical experiments in this section on a collection of 9 benchmark stochastic programming models from Ariyawansa and Felt (2004). We have specifically chosen these models due to the fact that they represented a diverse collection of stochastic optimization problems. On one hand, the models differ in the size of the instances, as well as the number of stages and the number of random variables in each stage. In addition, the models also pertain to decision-making problems across a wide range of application areas such as energy, finance, and telecommunications.



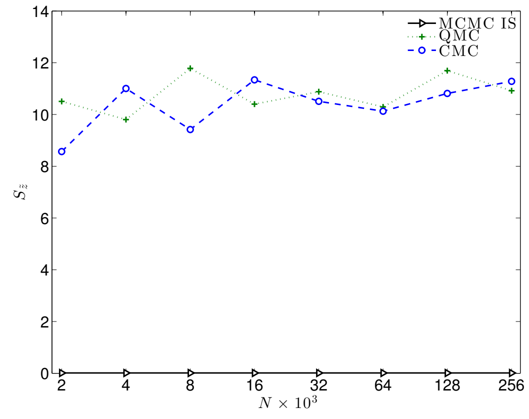
(a)



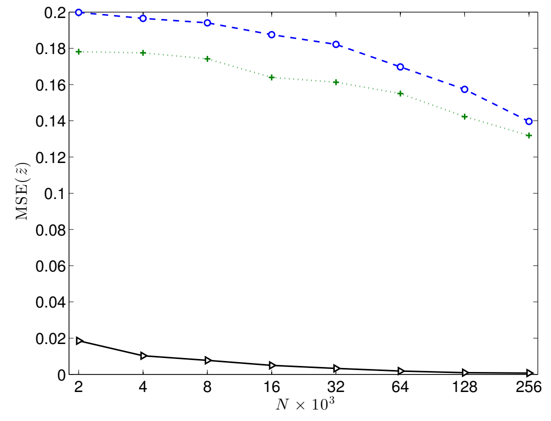
(b)



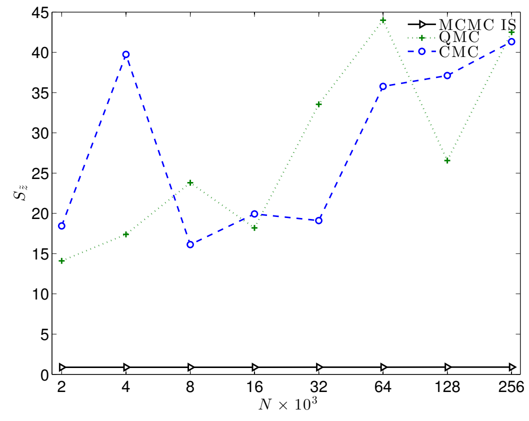
(c)



(d)



(e)



(f)

Figure 7 Error and variance of estimates for a Newsvendor problem where the uncertainty in demand and sales price is modeled using a lower-variance lognormal distribution with $\sigma = 1$ (7(a) - 7(b)), a higher-variance lognormal distribution with $\sigma = 2$ (7(c) - 7(d)), and multimodal rare-event distribution (7(e) - 7(f))

Problem	# Stages (T)	# Random Variables ($\sum_t D_t$)
Airlift Operation Scheduling (ASO)	2	2
Forest Planning (FP)	7	7
Electrical Investment (EI)	2	10
Selecting Currency Options (SCO)	4	4
Financial Planning Model (FPM)	2	16
Design of Batch Chemical Plants (DBCP)	2	4
Energy and Environmental Planning (EEP)	2	16
Telecommunications Network Planning (TNP)	2	15
Bond Investment Problem (BIP)	5	12

Table 1 Overview of the Test Problems from Ariyawansa and Felt (2004)

It is worth noting that many of the problems in Ariyawansa and Felt (2004) had to be modified in order to be solved with a sampling-based approach. This was due to the fact that many problems were originally formulated using discrete distributions and scenario trees (sometimes with 3 scenarios). In adapting these problems, we sought to change them as little as possible, and have therefore replaced each discrete distribution with a closely matching continuous distribution whose variance could be tuned. It is also worth noting that some problems were also formulated using integer variables. There have been efforts to extend the SDDP framework to allow for integer variables however such an extension is beyond the scope of the present paper. As such we have simply focused on solving the integer relaxations for these problems. Lastly, we note that we have omitted the "Cargo Network Scheduling" problem as it required the use of a non-linear programming solver. The full details of our modifications are listed in Appendix A.

5.2. Details on the Numerical Experiments

As in Section 4.8, we solved each of the models using the SDDP algorithm and compared the estimated optimal value $\tilde{z}$ when sampled cuts were generated using MCMC-IS, CMC and QMC.

We used $M = 3000$ samples to construct an approximate zero-variance importance sampling distribution in all of our experiments, and varied the number of samples to construct the sampled cut from $N = 2000$ to $N = 256000$. As before, we have ensured that all sampling methods were allotted an equal number of functional evaluations. In other words, the sampled cuts for CMC and QMC were constructed using $M + M_r + N$ total samples, where M_r denotes the number of rejected samples from the MCMC algorithm in MCMC-IS.

In the following experiments, we paired the SDDP algorithm with the stopping rule proposed in Shapiro (2011). This stopping rule terminates the SDDP algorithm as soon as the upper confidence bound $\bar{\theta} + z_{\alpha/2}\hat{\sigma_\theta}\sqrt{N}$ and the lower bound $\underline{\theta_k}$ is less than a prescribed tolerance level $\epsilon > 0$. In our experiments, we have set $\alpha = 5\%$ and $\epsilon = 10\%$. This means that we obtain a solution which achieves an value that is within 10% of the optimal value with 95% confidence.

In order to report error statistics as in the previous section, we have true optimal value for each model by solving each problem using the SDDP algorithm paired with the QMC method and an extremely large number of samples ($N = 10^7$). Such a large simulation is impractical in practice, but it was required to validate the correctness of the different methods. Of course we have no way of knowing that solutions obtained with $N = 10^7$ samples is the correct one, but all three algorithms converged to values that were within 1% of each other. As before, we have computed sample average values for all of our reported statistics using a total of 30 simulations, and have normalized all reported statistics for the sake of clarity.

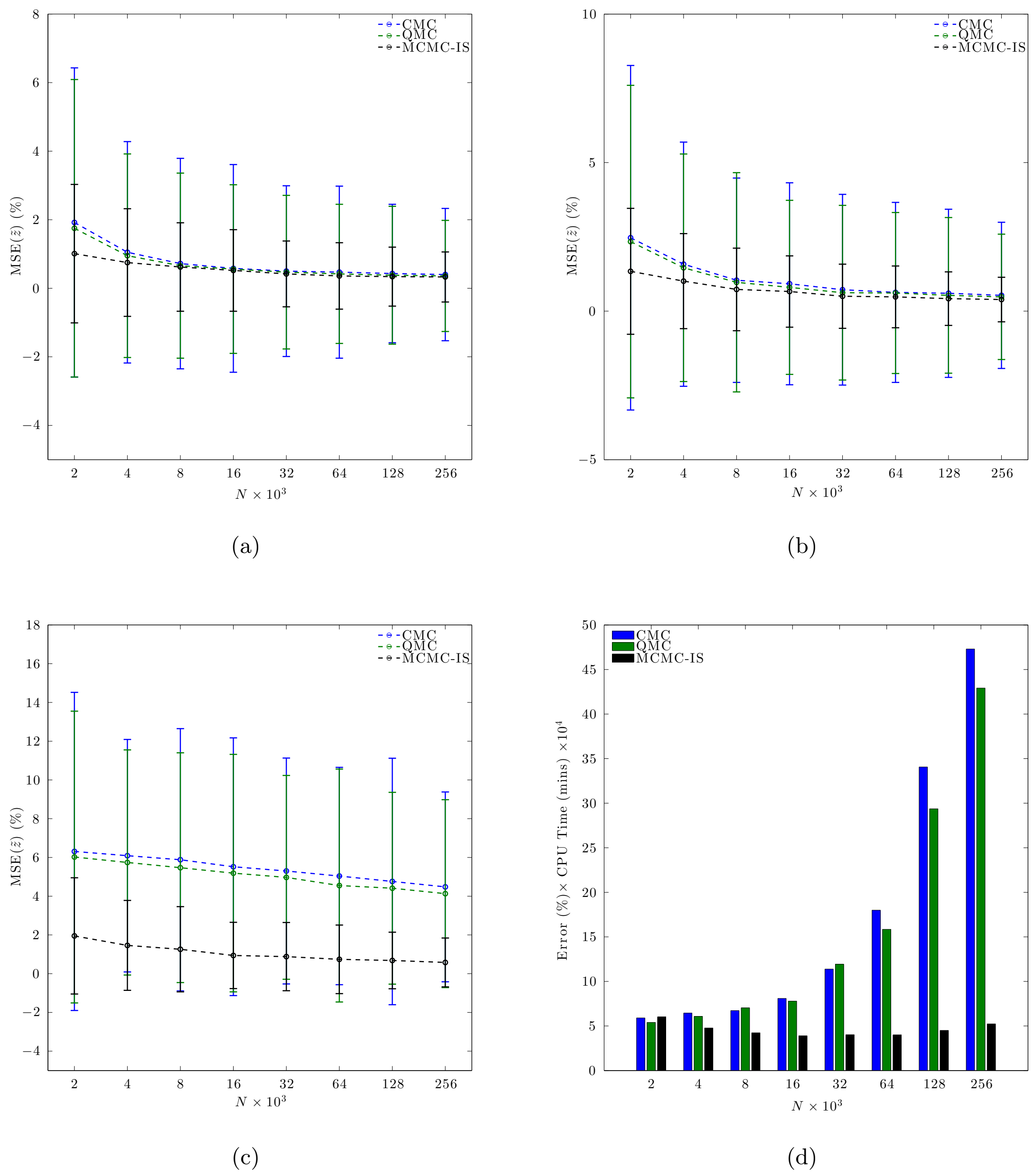


Figure 8 Average results with the Collection of Test Problems. (a) MSE($\tilde{z}$) for models with lower-variance distributions (b) MSE($\tilde{z}$) for models with higher-variance distributions (c) MSE($\tilde{z}$) for models with rare-event distribution. The error bars indicate the standard error associated with the solution obtained. (d) Error (%) $\times$ CPU Time (mins); for this plot we averaged the low variance, moderate variance and rare event results.

5.3. Accuracy and Variance of the Estimates

In Figure 8, we provide a summary of the error and sample standard deviation of the optimal value from the nine models when they are solved using MCMC-IS, QMC, and CMC methods. More specifically, these plots we show the average error and sample standard deviation for different sample sizes when the models contain lower-variance distributions (Figure 8(a)), higher-variance distributions (Figure 8(b)) and rare-event multimodal distributions (Figure 8(c)). We have plotted the average error across all 9 test problems. Given that the average values across different problems may be deceiving, we have also included a full table of these results for each problem and each value of N in Appendix B. Nevertheless, these results are consistent across different test problems, some of which are multistage, and have a markedly different structure.

When the models contain lower-variance distributions (Figure 8(a)), we see that all methods have low error (less then 5% in all cases) but that MCMC-IS estimates have lower variance. For models with higher-variance distributions (Figure 8(b)), MCMC-IS significantly outperforms the other methods, as MCMC-IS estimates of the optimal value have less error and less variance. This is also the case when models contain rare event distributions. In this case, MCMC-IS is the only method that can produce estimates near the true values using fewer than $N = 256000$ samples; the other two sampling methods exhibit an extremely slow convergence to the optimal value and require a far greater number of samples in order to converge.

In Figure 8(d) we plot the error times CPU time for each method (in % error $\times$ CPU time(min)). This metric provides insights into the relative "efficiency" of the different methods as it balances the conflicting requirements of obtaining highly accurate results using the least amount of CPU time. From the results in Figure 8(d) it can be seen that when the sample size is small (e.g. $N = 2000$), our method performs similarly to the other methods. This is because the advantage of error reduction comes at a high computational cost relative to the amount of time required to generate a small sample using CMC or QMC. When the sample size is larger (e.g $N = 8000$ and onwards), we see that the cost of MCMC-IS relative to other methods (while taking into consideration the error reduction) is much less.

6. Conclusions

Multistage stochastic programming models are considered computationally challenging mainly because the evaluation of the recourse function involves the solution of a multidimensional integral. Numerical methods such as Sample Average Approximation (SAA) and Stochastic Dual Dynamic Programming (SDDP) rely on sampling algorithms to approximately estimate the recourse function. The sampling algorithm used in conjunction with the optimization algorithm has a major bearing on the efficiency of the overall algorithm and on the accuracy of the solution. As a result the development of efficient sampling methods is an active area of research in stochastic programming.

The main contribution of this paper is the development of an importance sampling framework that is based on Markov Chain Monte Carlo (MCMC) to generate biased samples, and a kernel density estimation method to compute the likelihood function. Importance sampling has been proposed before in the literature of stochastic programming. The proposed method makes fewer restrictive assumptions than the importance sampling algorithm proposed in Dantzig and Glynn (1990) and Infanger (1992), and in particular can perform well even when the objective function is not additively separable. Our numerical experiments show that the method outperforms Crude Monte Carlo and Quasi Monte Carlo algorithms when the problem has moderate or high variance, and when the probability density function is difficult to sample from.

The results from numerical experiments suggest that MCMC-IS yields accurate estimates for models with lower-variance distributions and that it distinct advantage over sampling methods such as CMC and QMC when models are equipped with higher-variance distributions or rare-event distributions. We have also implemented the importance sampling technique from Infanger (1992) and in most cases it did not converge or was worse than CMC. We believe that the method proposed in Infanger (1992) is suitable for problems with a particular structure and may need further tuning for different test problems. Finally, it is clear from our results that if the stochastic program has rare events then the proposed method is the only one (from the ones we tested) that can produce reliable results. This last conclusion was not a surprise to us given that the MCMC method is known to perform well in such cases.

The importance sampling framework proposed in this paper could be extended in many ways. We have shown how importance sampling can be used in the context of a decomposition algorithm and expected value optimization. However, it is possible to use our approach with different algorithms (e.g. SAA) and with different types of stochastic programming models (e.g. risk averse stochastic programming). In addition, we have shown that the proposed method performs well when compared to existing methods.

Appendix A: Description of the Test Problems from Section 5

In this section, we explain the modifications we made to the set of test problems from Ariyawansa and Felt (2004).

Airlift Operation Scheduling (ASO): The aim of this model is to determine the optimal scheduling of several types of aircraft over different routes. Each aircraft type can only fly a certain number of hours within a month. Decision makers are allowed to switch aircraft from one route to another, under the condition that the switching hours from aircraft type i and from route j cannot exceed the original schedule. The objective is to minimize the cost of flights while ensuring that aircrafts can carry enough number of goods required for every route j. The demand of goods in route j is uncertain. Hence decision makers have to constantly take recourse actions in order to meet the actual requirement.

This is a stochastic programming model with $T = 2$ periods. The demand for route 1 is assumed to follow a lognormal distribution: $d_1 = 1000 \times \exp(0.1 \times \xi_1)$, where $\xi_1 \sim N(0, \sigma_i^2)$. The demand for route 2 is also log-normal: $d_2 = 1700 \times \exp(0.1 \times \xi_2)$, where $\xi_2 \sim N(0, \sigma_i^2)$. The coefficients 1000 and 1700 are chosen such that the demand given by these equations are as close as possible to the original problem. In particular, the mean demand for route i should be around 1000 and for route j should be around 1700. The value 0.1 is selected so that the uncertainty in demand does not vary too much.

In our experiments, we vary the amount of uncertainty in our model by setting $\sigma_i = i$. For experiments in which we pair this model with a rare-event distribution, we have spread the outcomes of the random variables across two important regions by using the transformation: $w(\xi) = \exp(\frac{\xi^2}{2} - \frac{(\xi+3)^2}{8}) + \exp(\frac{\xi^2}{2} - \frac{(\xi+1)^2}{8})$. As a result, the recourse function $Q(x, \xi_1, \xi_2)f(\xi_1, \xi_2)$ is replaced with $Q(x, w(\xi_1), w(\xi_2))f(\xi_1, \xi_2)$. All other information such as the flying hours per trip, carrying capacity, cost per flight, penalty costs, flying hours after switching flights, and the cost per flight switched are kept the same as the original test problem.

Forest Planning (FP): The aim of this model is to decide how to harvest a forest in order to maximize the final value of timber that obtain after T stages. To model this problem, the forest area is divided into $K = 8$ segments according to the ages of trees. In any period, trees that are not harvested or destroyed by fire will be transferred to the next age class. In addition, forest planners have to decide how much area in each age class will be harvested while minimizing the risk that a random proportion of the remained forest could be destroyed by fire. It is assumed that the burned areas will quickly get replaced and started at age class 1. As each period of time can last for 20 years, the future value of timber will be discounted at a given rate, δ.

This is a stochastic programming model with $T = 7$ periods. Instead of discretizing the fire rate as in the original problem, the fire rate is now described by $0.07 \times \exp(0.1 \times \xi_j)$, where $\xi_j \sim N(0, \sigma_i^2)$ and $j = 1, ..., K$. The value of 0.07 is chosen so that the fire rate is as close as possible to the values given in the original test problem. In our experiments, we vary the amount of uncertainty in our model by setting $\sigma_i = i$. For experiments in which we pair this model with a rare-event distribution, we have spread the outcomes of the random variables across two important regions by using the transformation: $w(\xi) = \exp(\frac{\xi^2}{2} - \frac{(\xi+3)^2}{8}) + \exp(\frac{\xi^2}{2} - \frac{(\xi+1)^2}{8})$.

All other information such as the number of age classes K, the initial forest area of each age class s_1, the discount rate δ, the value of standing timber v, the yields of harvest y and two constants α, β that limit the change in purchasing timber from one period to the next are all kept the same as the original formulation.

Electrical Investment (EI): In this model, decision makers have to decide how much to invest in $n = 4$ different power system technologies in order to produce electricity. Each technology has a random investment cost, operating cost, and an availability factor corresponding to the time during which each technology operates. The objective is to minimize the total cost while satisfying the electricity demand. The demand of electricity is uncertain and is modeled as different modes in the load duration curve. There is a penalty charge if there is a shortage of electricity production. In addition, there is a limitation on how much the producers can invest to expand the electricity supply at every period of time.

This is a stochastic programming model with $T = 2$ periods. To make the model more realistic, we have increased the number of intervals (modes) used to create the load duration curve into a large number of intervals from 3 modes to 10 modes. We have set the demand for each mode as:

$$\begin{aligned}
d_1 &= 5 \times \exp(0.1 \times \xi_1)\\
d_2 &= 20 \times \exp(0.1 \times \xi_2)\\
d_3 &= 20 \times \exp(0.1 \times \xi_3)\\
d_4 &= 15 \times \exp(0.1 \times \xi_4)\\
d_5 &= 10 \times \exp(0.1 \times \xi_5)\\
d_6 &= 8 \times \exp(0.1 \times \xi_6)\\
d_7 &= 8 \times \exp(0.1 \times \xi_7)\\
d_8 &= 4 \times \exp(0.1 \times \xi_8)\\
d_9 &= 5 \times \exp(0.1 \times \xi_9)\\
d_{10} &= 5 \times \exp(0.1 \times \xi_{10})
\end{aligned}$$

where $\xi_1, \ldots, \xi_{10} \sim N(0, \sigma_i^2)$. In our experiments, we vary the amount of uncertainty in our model by setting $\sigma_i = i$. For experiments in which we pair this model with a rare-event distribution, we have spread the outcomes of the random variables across two important regions by using the transformation: $w(\xi) = \exp(\frac{\xi^2}{2} - \frac{(\xi+3)^2}{8}) + \exp(\frac{\xi^2}{2} - \frac{(\xi+1)^2}{8})$.

We note that we have chosen the coefficients for the load blocks described above so as to create a smooth and realistic load duration curve. We assume that the operating costs of mode 2 are 90% of the operating costs in mode 1. Given that the operating costs of mode 3 should be smaller than the operating costs of mode 2, we have set these to 85%. Following this pattern, operating costs of each subsequent mode decrease by 5%. Lastly, given that the the demand has increased from the original value of 12 to 100, we have also increases the total investment budget from 120 to 1200.

Selecting Currency Options (SCO): Many multinational corporations have a substantial amount of revenue across a wide number of different currencies. If the foreign exchange rate decreases, the actual revenue received will be less than predicted. To hedge this risk, decision makers can purchase currency options, which guarantees a certain exchange rate (also known as strike price) at some point in the future.

The aim of this model is to help corporates minimize the cost of purchasing currency options while ensuring that their payoff is greater than a level specified by the company. This level is normally known as the target exchange rate. The random variables in this model are the exchange rate and option prices at time t. The number of time periods is four. The interest rate is set to 0.10 and the volatility of the exchange rate is set to 0.11 throughout all of time periods. The number of options is 10 with strike prices as follows: $E_1 = 0.44; E_2 = 0.50; E_3 = 0.57; E_4 = 0.63; E_5 = 0.70; E_6 = 0.76; E_7 = 0.83; E_8 = 0.89; E_9 = 0.96; E_{10} = 1.02$. In order to simplify the problem, foreign interest rate is set to the UK interest rate of 0.5%. The exchange rate is given by $S = 0.5 * \exp(0.2 * \xi)$, where $\xi \sim N(0, \sigma_i^2)$. The coefficient 0.5 and 0.2 are chosen such that the generated exchange rate are as close as possible to the original source. Finally, the target exchange rate is set to 0.463, which is the average of target exchange rates across all scenarios shown in Table 5 in Ariyawansa and Felt (2004). In our experiments, we vary the amount of uncertainty in our model by setting $\sigma_i = i$. For experiments in which we pair this model with a rare-event distribution, we have spread the outcomes of the random variables across two important regions by using the transformation: $w(\xi) = \exp(\frac{\xi^2}{2} - \frac{(\xi+3)^2}{8}) + \exp(\frac{\xi^2}{2} - \frac{(\xi+1)^2}{8})$.

Financial Planning Model (FPM): The aim of this model is to maximize the expected value of savings and general accounts while avoiding the shortfall in these accounts. Due to the structures and regulations of different types of insurance policies, there are several constraints in the problem. Also, different types of investment (i.e. direct or indirect) may result in different calculations. The random variables are: income return, price return, interest rate, deposit inflow, principle payments, interest payments and total reserve liability. The number of time periods is two and the number of funds we used is five. Since the data are not given precisely either in the original source or in Ariyawansa and Felt (2004), the data are created such that they are as close as possible to some of the examples found in the original paper. The data are generated as follows:

$$\begin{aligned}
RI_{(nt+1)} &= 0.15 \times \exp(0.1 \times \xi_1) \\
RP_{(nt+1)} &= 0.20 \times \exp(0.1 \times \xi_2) \\
g_{t+1} &= 0.05 \times \exp(0.1 \times \xi_3) \\
F_{t+1} &= 200 \times \exp(0.1 \times \xi_4) \\
P_{t+1} &= 400 \times \exp(0.1 \times \xi_5) \\
I_{t+1} &= 75 \times \exp(0.1 \times \xi_6) \\
L_t &= 700 \times \exp(0.1 \times \xi_7)
\end{aligned}$$

where $\xi_1, \ldots, \xi_7 \sim N(0, \sigma_i^2)$. We kept the same notation as in Ariyawansa and Felt (2004) and the explanation of what each variable means can be found there. The only piece of information that cannot be found is the income gap IG_{t+1}. In this case, we propose to describe the income gap as the percentage given by the equation $1 + 0.3 \times \exp(0.1 * \xi_8)$, where $\xi_8 \sim N(0, \sigma_i^2)$. This number will be multiplied with the value of funds to find the investment income. n our experiments, we vary the amount of uncertainty in our model by setting $\sigma_i = i$. For experiments in which we pair this model with a rare-event distribution, we have spread the outcomes of the random variables across two important regions by using the transformation: $w(\xi) = \exp(\frac{\xi^2}{2} - \frac{(\xi+3)^2}{8}) + \exp(\frac{\xi^2}{2} - \frac{(\xi+1)^2}{8})$.

Design of Batch Chemical Plants (DBCP): The aim of this model is to decide what kinds of chemical plants should be built in order to maximize the profit from selling chemical products - all the while minimizing the investment costs for these plants. Decision makers have to what kinds of chemical plants to build, as well as how many of them, and how they should be built. Decision makers also have to decide which tasks to perform on a particular plant while satisfying the constraint of capacity, processing time of each task and the limited amount of resource.

This is a stochastic programming model with $T = 2$ periods. The random variables in this problem are: the demand and the price per unit mass of resource. The random variables are changed from discrete to continuous distribution as follows:

- the demand for resource 4 is: $Q_4 = 150 \times \exp(0.5 \times \xi_1)$.
- the price of resource 4 is: $v_4 = 55 \times \exp(0.1 \times \xi_2)$.
- the demand for resource 7 is: $Q_7 = 200 \times \exp(0.5 \times \xi_3)$.
- the price of resource 7 is: $v_7 = 80 \times \exp(0.1 \times \xi_4)$

where $\xi_1, \ldots, \xi_4 \sim N(0, \sigma_i^2)$. In our experiments, we vary the amount of uncertainty in our model by setting $\sigma_i = i$. For experiments in which we pair this model with a rare-event distribution, we have spread the outcomes of the random variables across two important regions by using the transformation: $w(\xi) = \exp(\frac{\xi^2}{2} - \frac{(\xi+3)^2}{8}) + \exp(\frac{\xi^2}{2} - \frac{(\xi+1)^2}{8})$. The coefficients are selected so that the quantities described by these equations are as close as possible to the data found in the original paper. All other information is the same as the original formulation.

Energy and Environmental Planning (EEP): The objective of this model is to minimize investment costs as well as operating costs of different types of energy technologies while making sure that the electricity production satisfies the demand of each utility. The model shows a great degree of realism by taking into account various aspects of the problem including the production constraints, capacity expansions constraints, equilibrium constraints, and environmental constraints. The energy supply and demand are classified by many different technologies. Depending on different types of technologies, there are different ways of calculating their productions and energy balances. The problem also considers the peak demand level and ensures that the capacity of the production can cover the peak demands. Hence there are peak demand constraints for different types of technologies. In addition, different technologies can perform at different levels depending on the season (i.e. winter or summer) and the time of day. The problem also takes into account the environmental aspects, in which the CO_2 level produced by all of these technologies has to be less than a certain level, which is uncertain in the future. Hence the random variable in this problem is the CO_2 limit.

Telecommunication Network Planning (TNP): There are many nodes in a telecommunication network. Between any two nodes, there are several possible routes to connect them together. At any time, there are various point-to-point pairs that need to be served by the network. This demand is random. The purpose of this model is to decide which links to connect within a communication network while minimizing the unserved requests and satisfying a budget constraint.

This is a stochastic programming model with with $T = 2$ time periods. Assuming that there is a huge increase in the demand for telecommunication in the future, the budget for network expansion should be set at a reasonably high level of the value of 5. This is equivalent to about 22% increase in the initial capacity of the network. The demand for every point-to-point pair i is given as: $d_j = 3 \times \exp(0.2 \times \xi_j)$, where $\xi_j \sim N(0, \sigma_i^2)$, where $j = 1, ..., 15$. In our experiments, we vary the amount of uncertainty in our model by setting $\sigma_i = i$. For experiments in which we pair this model with a rare-event distribution, we have spread the outcomes of the random variables across two important regions by using the transformation: $w(\xi) = \exp(\frac{\xi^2}{2} - \frac{(\xi+3)^2}{8}) + \exp(\frac{\xi^2}{2} - \frac{(\xi+1)^2}{8})$.

Bond Investment Problem (BIP): The objective of this model is to maximize the expected return on bond lending and the balance of transactions while minimizing the cost of bond borrowing. In this model, the random variables are the rates of return on bond lending, bond borrowing and total balance of transactions, as well as the growth rate of the transactions.

This is a stochastic programming model with $T = 5$ periods. In each period, there is a limited number of bonds that can be traded. The rate of return on lending is described by $0.07 \times \exp(0.1 \times \xi_1)$. The rate of borrowing is given as $0.1 \times \exp(0.1 \times \xi_2)$, and the rate of return on balance transaction is given as $0.15 \times \exp(0.1 \times \xi_3)$ where $\xi_1, \ldots, \xi_3 \sim N(0, \sigma_i^2)$. This corresponds to the rate of return on lending of around 7%, rate of borrowing of around 10% and rate of return on the balance of transactions of around 15% with a reasonable fluctuation in their quantities. We assume that the total balance of bond transactions increased by a stochastic quantity $\xi_t = p \times \xi_{t-1}$, where $p = 0.1 \times \exp(0.1 \times \xi_3), \xi_3 \sim N(0, \sigma_i^2)$. This is equivalent to the increase of around 10% increase (with a certain amount of uncertainty) in the balance of transactions in every time period.

In our experiments, we vary the amount of uncertainty in our model by setting $\sigma_i = i$. For experiments in which we pair this model with a rare-event distribution, we have spread the outcomes of the random variables across two important regions by using the transformation: $w(\xi) = \exp(\frac{\xi^2}{2} - \frac{(\xi+3)^2}{8}) + \exp(\frac{\xi^2}{2} - \frac{(\xi+1)^2}{8})$.

Appendix B: Detailed Numerical Results from Section 5

Table 2: Comparison of performance of MC, QMC and the proposed algorithm (MCMC) when the random variables are drawn from the distribution with standard deviation of 1

Problem	# Samples (N)	MSE($\tilde{z}$) (%)			SD($\tilde{z}$) (%)			# of Iterations			Runtime (s)		
		MC	QMC	MCMC	MC	QMC	MCMC	MC	QMC	MCMC	MC	QMC	MCMC
AOS	2000	1.25	1.22	0.87	3.95	3.86	2.02	18	21	4	2	2	8
	4000	0.76	0.71	0.69	3.91	3.88	1.97	20	22	6	2	4	9
	8000	0.54	0.53	0.48	3.89	3.89	1.32	24	23	6	2	5	12
	16000	0.42	0.45	0.41	3.91	3.90	2.05	25	26	5	7	7	13
	32000	0.39	0.37	0.36	3.91	3.90	2.12	26	25	6	10	12	17
	64000	0.34	0.35	0.31	3.93	3.91	1.97	28	27	8	19	22	22
	128000	0.38	0.32	0.28	3.92	3.91	2.17	29	30	14	73	78	52
	256000	0.32	0.29	0.27	3.91	3.92	1.97	33	32	16	138	137	86
FP	2000	0.98	0.89	0.76	5.12	5.04	2.36	92	92	62	67	87	1106
	4000	0.61	0.58	0.55	5.11	5.03	2.31	93	92	64	79	150	1141
	8000	0.49	0.45	0.47	5.07	5.02	2.27	94	93	63	93	176	1134
	16000	0.35	0.34	0.35	5.06	5.01	1.68	112	101	65	272	249	1540
	32000	0.32	0.32	0.28	5.06	5.01	2.26	118	112	69	420	536	1779
	64000	0.33	0.28	0.27	5.08	5.02	1.33	125	118	72	836	876	1938
	128000	0.30	0.29	0.27	5.07	5.01	2.32	131	125	76	2998	2963	2468
	256000	0.31	0.27	0.26	5.03	5.01	2.06	134	130	78	4987	5029	3751
EI	2000	1.56	1.54	0.88	3.99	3.97	2.07	19	21	5	2	3	7
	4000	0.78	0.73	0.67	4.01	3.98	2.03	25	22	5	4	4	11
	8000	0.52	0.51	0.32	4.01	3.97	2.04	22	24	5	5	6	14
	16000	0.47	0.43	0.46	3.99	3.99	2.12	25	26	6	9	10	8
	32000	0.36	0.33	0.32	3.97	3.96	2.01	24	26	7	16	18	16
	64000	0.45	0.32	0.33	3.99	3.96	1.95	29	25	11	33	31	29
	128000	0.38	0.31	0.28	3.96	3.95	2.03	28	29	14	65	71	55
	256000	0.34	0.28	0.28	3.98	3.93	2.02	33	30	18	153	147	75
SCO	2000	2.51	2.14	1.27	3.94	3.93	2.83	66	63	53	373	359	1008
	4000	1.24	1.08	0.78	3.95	3.92	1.76	67	63	58	446	363	1160
	8000	0.91	0.83	0.85	3.93	3.90	2.86	70	65	56	481	457	963
	16000	0.48	0.49	0.54	3.91	3.91	2.42	72	67	61	526	502	1159

	32000	0.42	0.44	0.41	3.92	3.90	2.06	79	73	63	705	694	1228
	64000	0.42	0.38	0.30	3.93	3.89	2.12	83	79	62	1070	1099	1394
	128000	0.39	0.34	0.32	3.93	3.89	1.85	89	85	68	1499	1483	1614
	256000	0.38	0.31	0.29	3.90	3.87	1.93	94	91	71	1860	1899	1798
FPM	2000	3.42	2.95	1.46	5.09	4.98	2.54	30	34	31	62	73	122
	4000	1.65	1.15	0.98	5.12	4.98	2.51	32	35	33	67	81	125
	8000	0.98	0.81	0.64	5.06	5.02	1.65	37	36	32	78	79	133
	16000	0.76	0.89	0.73	5.06	4.97	1.97	41	49	36	92	114	147
	32000	0.73	0.61	0.61	4.98	4.93	2.56	45	49	48	110	130	188
	64000	0.62	0.53	0.63	5.08	4.92	2.39	58	62	49	158	170	209
	128000	0.52	0.45	0.37	4.98	4.93	1.85	65	73	51	227	254	239
	256000	0.47	0.41	0.48	4.97	4.95	2.46	72	74	47	305	321	258
DBCP	2000	1.67	1.55	0.72	3.82	3.78	1.92	26	28	16	36	39	77
	4000	0.94	1.03	0.61	3.79	3.76	1.95	28	29	20	45	43	77
	8000	0.65	0.62	0.47	3.81	3.76	2.08	32	33	22	54	57	97
	16000	0.45	0.53	0.46	3.81	3.79	2.02	34	37	23	62	68	108
	32000	0.49	0.43	0.36	3.82	3.73	1.96	39	40	25	88	94	122
	64000	0.38	0.36	0.26	3.83	3.75	1.87	49	44	28	156	144	149
	128000	0.43	0.32	0.32	3.81	3.72	1.67	56	55	32	229	236	187
	256000	0.41	0.31	0.28	3.82	3.69	1.62	57	59	34	281	301	215
EEP	2000	1.45	1.32	0.82	2.63	2.61	1.56	34	32	26	20	12	74
	4000	0.72	0.71	0.64	2.63	2.61	1.59	39	38	28	28	25	83
	8000	0.59	0.55	0.52	2.64	2.59	1.62	37	38	29	31	35	88
	16000	0.65	0.52	0.43	2.61	2.60	1.38	44	42	33	40	43	105
	32000	0.46	0.48	0.38	2.58	2.58	1.71	55	57	35	82	97	138
	64000	0.51	0.41	0.28	2.59	2.56	1.78	64	66	38	144	132	160
	128000	0.41	0.38	0.35	2.59	2.57	1.56	65	67	42	234	196	209
	256000	0.40	0.38	0.30	2.58	2.55	1.43	68	69	41	313	297	242
TNP	2000	2.54	2.33	0.96	3.97	3.93	2.18	19	21	18	9	19	42
	4000	1.34	1.21	0.89	4.02	3.92	1.92	22	22	18	11	20	47
	8000	0.96	0.84	0.69	3.96	3.92	1.94	25	24	20	14	28	50
	16000	0.57	0.47	0.47	3.96	3.91	2.19	29	28	22	22	37	60
	32000	0.54	0.45	0.45	3.94	3.82	1.95	31	32	25	29	48	80
	64000	0.41	0.37	0.36	3.88	3.83	1.19	32	30	25	43	66	84
	128000	0.38	0.36	0.35	3.86	3.83	1.58	38	29	26	94	87	112

	256000	0.40	0.35	0.34	3.85	3.83	1.88	40	39	32	151	165	133
BIP	2000	2.87	2.79	1.37	5.16	4.57	2.98	78	79	64	234	165	694
	4000	1.39	1.33	0.97	5.19	4.51	2.87	81	80	67	294	273	747
	8000	0.88	0.82	1.14	5.20	5.09	2.32	83	81	68	356	378	856
	16000	1.03	0.92	0.87	5.16	5.06	1.17	88	85	71	415	443	929
	32000	0.78	0.81	0.63	5.18	5.08	2.55	92	87	75	695	745	1169
	64000	0.74	0.74	0.46	5.16	5.06	2.43	96	91	78	1150	922	1461
	128000	0.64	0.62	0.51	5.08	4.98	2.78	101	96	82	1841	1413	1628
	256000	0.61	0.60	0.51	5.09	4.98	2.35	106	103	87	2439	2296	1749

Table 3: Comparison of performance of MC, QMC and the proposed algorithm (MCMC) when the random variables are drawn from the distribution with standard deviation of 2

Problem	# Samples (N)	MSE($\tilde{z}$) (%)			SD($\tilde{z}$) (%)			# of Iterations			Runtime (s)		
	per period	MC	QMC	MCMC	MC	QMC	MCMC	MC	QMC	MCMC	MC	QMC	MCMC
AOS	2000	2.03	1.92	0.96	3.77	3.62	1.47	21	24	7	2	3	16
	4000	1.17	1.05	0.87	3.68	3.66	1.32	22	26	6	3	5	15
	8000	0.84	0.73	0.68	3.78	3.75	2.03	23	25	7	7	8	20
	16000	0.62	0.57	0.63	3.77	3.76	1.72	27	28	8	10	12	27
	32000	0.67	0.52	0.52	3.88	3.75	1.08	26	29	8	16	20	24
	64000	0.57	0.52	0.48	3.76	3.77	2.03	30	30	12	39	39	36
	128000	0.52	0.43	0.32	3.76	3.78	1.72	30	30	14	72	77	87
	256000	0.48	0.40	0.31	3.77	3.77	1.52	34	30	17	168	149	139
FP	2000	1.19	1.08	0.79	5.94	5.92	2.45	96	93	67	118	131	1214
	4000	0.68	0.62	0.53	5.93	5.91	1.52	98	94	66	131	135	1175
	8000	0.53	0.52	0.38	5.92	5.89	2.38	101	96	69	142	150	1332
	16000	0.51	0.49	0.43	5.92	5.90	2.14	118	107	69	316	293	1526
	32000	0.48	0.46	0.36	5.91	5.90	1.98	122	114	73	473	477	1929
	64000	0.47	0.44	0.40	5.91	5.91	2.23	128	121	75	914	994	2043
	128000	0.40	0.42	0.37	5.93	5.88	2.04	137	126	77	3139	2940	3319
	256000	0.42	0.41	0.35	5.91	5.89	1.67	142	136	78	5444	5387	4567
EI	2000	2.12	2.09	0.96	4.12	4.10	1.96	22	23	7	3	3	15
	4000	1.05	0.98	0.85	4.12	4.09	1.45	27	25	7	5	5	17
	8000	0.89	0.73	0.57	4.11	4.07	1.78	27	25	8	7	6	23
	16000	0.78	0.66	0.54	4.10	4.09	1.56	27	27	9	11	12	29
	32000	0.85	0.61	0.57	4.09	4.05	1.86	30	29	8	20	21	24
	64000	0.72	0.71	0.52	4.09	4.07	1.93	33	31	13	44	43	43
	128000	0.67	0.62	0.52	4.09	4.08	1.45	35	34	17	88	90	105
	256000	0.62	0.61	0.49	4.08	4.09	1.86	38	37	20	171	167	173
SCO	2000	4.14	4.07	2.69	4.27	4.22	1.84	69	67	56	402	403	1196
	4000	2.34	2.28	1.65	4.27	4.21	1.95	70	67	59	450	455	1221
	8000	1.61	1.36	1.04	4.25	4.20	1.56	73	68	59	494	484	1428
	16000	1.36	1.04	0.95	4.26	4.22	1.74	75	69	63	569	538	1598
	32000	0.74	0.63	0.41	4.27	4.23	1.45	83	74	64	767	726	1664
	64000	0.58	0.72	0.53	4.28	4.22	1.75	89	82	65	1164	1094	1772

	128000	0.56	0.52	0.41	4.27	4.23	1.67	92	84	70	1771	1631	1976
	256000	0.54	0.48	0.32	4.26	4.21	1.78	98	93	73	2754	2624	2125
FPM	2000	4.23	4.17	1.84	5.87	5.82	2.01	32	36	33	82	96	202
	4000	2.43	2.34	1.42	5.85	5.81	1.96	34	35	34	91	98	217
	8000	1.48	1.43	0.82	5.83	5.74	2.23	39	38	36	108	106	235
	16000	0.98	0.93	0.84	5.85	5.73	2.08	42	42	39	122	135	272
	32000	0.72	0.64	0.60	5.84	5.76	2.15	46	47	45	148	165	365
	64000	0.74	0.62	0.54	5.83	5.73	2.05	62	63	51	328	331	479
	128000	0.67	0.62	0.53	5.80	5.74	2.07	69	71	55	489	489	565
	256000	0.54	0.52	0.45	5.78	5.73	2.10	81	82	58	691	718	689
DBCP	2000	2.57	2.34	0.98	4.63	4.65	1.97	28	27	23	61	60	141
	4000	1.34	1.23	0.79	4.62	4.63	1.93	31	28	26	72	68	164
	8000	0.87	0.83	0.72	4.63	4.62	2.07	33	33	25	76	84	167
	16000	0.86	0.65	0.56	4.61	4.63	1.87	36	35	27	109	145	187
	32000	0.62	0.62	0.53	4.61	4.66	2.03	40	38	28	172	189	177
	64000	0.73	0.57	0.41	4.63	4.68	2.18	49	47	35	311	361	250
	128000	0.62	0.53	0.42	4.65	4.68	1.65	57	54	39	443	427	312
	256000	0.57	0.51	0.41	4.64	4.64	1.61	61	60	45	573	574	455
EEP	2000	1.94	1.72	1.04	3.04	3.03	1.48	37	36	28	34	35	109
	4000	1.12	1.05	0.85	3.06	3.03	1.46	42	40	29	50	49	113
	8000	0.73	0.64	0.72	3.05	2.98	1.52	43	41	31	53	55	130
	16000	0.64	0.47	0.57	3.06	3.02	1.43	47	44	36	63	74	158
	32000	0.64	0.42	0.45	3.06	2.99	1.48	55	52	37	111	127	185
	64000	0.52	0.47	0.41	3.05	2.98	1.64	63	61	42	276	298	221
	128000	0.51	0.45	0.37	3.07	2.97	1.51	69	69	48	566	609	348
	256000	0.40	0.38	0.35	3.07	2.98	1.42	72	73	53	817	840	483
TNP	2000	3.80	3.56	1.32	4.53	4.47	1.97	22	21	20	22	28	66
	4000	2.03	1.73	0.93	4.56	4.46	2.05	25	23	21	25	34	75
	8000	1.04	0.86	0.75	4.54	4.45	1.83	27	24	20	33	48	76
	16000	0.65	0.64	0.62	4.55	4.43	1.95	30	29	22	44	65	88
	32000	0.73	0.53	0.47	4.56	4.45	2.35	34	33	23	57	85	109
	64000	0.61	0.62	0.42	4.57	4.45	1.75	37	35	27	85	98	141
	128000	0.52	0.49	0.43	4.57	4.46	1.71	42	38	30	183	182	164
	256000	0.52	0.52	0.45	4.56	4.45	1.63	46	44	35	298	286	252
BIP	2000	4.32	4.18	1.84	5.68	5.57	2.05	83	81	68	344	383	749

4000	2.18	2.03	1.21	5.66	5.58	2.13	85	82	68	361	395	781
8000	1.39	1.25	0.92	5.67	5.63	2.36	85	83	70	406	424	887
16000	1.21	1.03	0.84	5.65	5.61	2.27	92	89	73	452	463	976
32000	0.72	0.75	0.62	5.67	5.62	2.04	95	94	76	783	825	1226
64000	0.67	0.58	0.61	5.66	5.58	1.97	102	98	81	1279	1247	1539
128000	0.65	0.52	0.40	5.67	5.63	2.01	108	104	86	2060	2141	1731
256000	0.62	0.48	0.36	5.67	5.59	1.94	116	112	93	2921	2827	1968

Table 4: Comparison of performance of MC, QMC and the proposed algorithm (MCMC) when the random variables are drawn from the rare-event distribution

Problem	# Samples (N)	MSE($\tilde{z}$) (%)			SD($\tilde{z}$) (%)			# of Iterations			Runtime (s)		
	per period	MC	QMC	MCMC	MC	QMC	MCMC	MC	QMC	MCMC	MC	QMC	MCMC
AOS	2000	5.62	5.34	1.32	8.78	8.48	2.03	23	23	9	8	10	43
	4000	5.36	4.87	0.96	8.48	8.59	2.03	23	24	9	9	18	47
	8000	4.54	4.21	0.82	8.35	8.47	1.90	25	24	9	10	19	54
	16000	3.95	3.66	0.76	8.28	8.37	1.47	25	26	14	26	29	109
	32000	3.59	3.43	0.63	8.24	8.26	1.76	27	26	15	42	55	133
	64000	3.70	3.24	0.58	8.25	8.27	1.90	31	28	17	89	84	153
	128000	3.45	3.16	0.43	8.22	8.18	1.83	33	32	19	332	335	233
	256000	2.98	2.84	0.42	8.19	8.17	1.54	35	33	22	588	590	357
FP	2000	7.85	7.23	1.82	9.66	9.32	2.79	98	96	64	289	364	3335
	4000	7.74	7.15	1.38	9.59	9.21	2.25	99	97	65	343	642	3438
	8000	7.62	6.93	1.23	9.53	9.02	2.31	103	99	66	424	746	3504
	16000	7.32	6.84	0.94	9.43	8.94	2.03	112	106	66	1090	1047	4676
	32000	6.92	6.65	0.89	9.34	9.03	2.94	120	114	69	1755	2165	5323
	64000	6.42	5.87	0.68	9.28	8.96	2.57	129	122	73	3460	3627	5847
	128000	6.28	5.72	0.66	9.22	9.10	1.96	134	128	76	12262	12136	7443
	256000	5.83	5.35	0.43	9.12	9.12	2.09	137	132	80	20399	20438	11564
EI	2000	6.43	6.32	2.03	8.82	8.76	2.12	24	24	13	9	10	55
	4000	6.26	6.17	1.22	8.71	8.82	2.07	28	27	15	19	17	68
	8000	6.20	6.02	0.94	8.83	8.63	1.97	27	28	14	19	21	73
	16000	5.92	5.53	0.93	8.96	8.69	2.15	28	28	17	31	30	101

	32000	5.63	5.38	0.84	8.95	8.52	2.02	31	31	18	43	52	110
	64000	5.22	4.65	0.62	8.91	8.76	2.08	33	32	21	89	87	151
	128000	4.49	4.28	0.64	8.89	8.42	2.10	36	33	26	190	199	218
	256000	4.34	4.08	0.52	8.87	8.75	2.14	39	36	28	415	393	300
SCO	2000	8.26	7.91	2.78	9.71	9.64	2.46	71	73	57	1610	1681	2971
	4000	8.36	7.62	1.88	9.62	9.54	2.35	73	72	59	1964	1667	3232
	8000	8.03	7.39	1.39	9.61	9.47	2.84	74	75	59	2046	2116	3417
	16000	7.65	7.23	0.93	9.62	9.81	2.17	77	76	62	2327	2291	3779
	32000	7.55	6.88	0.97	9.67	9.38	2.54	85	81	65	3070	3157	4232
	64000	7.21	6.43	0.78	9.74	9.82	2.35	92	86	68	4752	4795	4573
	128000	6.84	6.21	0.65	9.68	9.76	2.27	96	90	71	6471	6303	5109
	256000	6.14	5.75	0.53	9.65	9.12	2.23	102	96	75	8083	8066	6200
FPM	2000	8.91	8.53	2.01	8.75	9.14	2.55	36	35	33	301	305	387
	4000	8.16	7.72	1.85	9.82	9.04	2.96	38	37	35	320	345	417
	8000	8.58	7.44	1.27	9.45	9.12	2.85	39	38	36	332	357	459
	16000	7.83	7.39	0.92	9.34	9.35	2.86	44	42	39	403	395	469
	32000	7.48	7.05	1.11	9.72	9.02	2.64	49	46	47	481	490	633
	64000	7.21	6.52	0.84	9.41	8.97	2.47	56	52	51	618	602	731
	128000	7.03	6.61	0.72	9.60	9.22	2.35	68	59	55	958	831	814
	256000	6.75	6.26	0.59	9.64	9.16	2.24	74	68	59	1255	1187	923
DBCP	2000	4.26	4.03	1.85	7.59	7.02	2.05	30	30	23	168	171	271
	4000	4.02	3.74	1.64	7.41	6.97	1.96	32	32	24	207	192	346
	8000	3.69	3.32	1.25	7.36	6.86	1.84	35	34	26	237	237	381
	16000	3.41	3.21	0.98	7.45	6.78	1.93	39	38	28	284	282	415
	32000	3.32	3.02	0.82	7.54	7.06	1.85	43	42	29	388	400	438
	64000	3.17	2.85	0.77	7.34	6.92	1.85	50	48	34	639	636	556
	128000	3.09	2.79	0.69	7.22	7.01	2.04	58	56	39	954	967	719
	256000	2.91	2.64	0.56	7.38	6.88	1.86	62	59	46	1222	1213	928
EEP	2000	3.45	3.21	1.73	6.87	6.32	2.13	39	37	29	90	57	248
	4000	3.23	3.03	1.52	6.42	6.26	2.42	42	41	28	123	110	250
	8000	3.19	2.95	1.47	6.56	6.26	2.04	43	40	31	151	153	284
	16000	3.06	2.82	1.12	6.77	6.33	2.45	48	47	35	180	214	379
	32000	2.91	2.56	0.97	6.64	6.28	2.10	55	52	38	331	358	448
	64000	2.72	2.43	0.73	6.70	6.31	2.36	63	59	39	576	529	503
	128000	2.63	2.34	0.79	6.51	6.35	2.22	70	66	45	1027	846	720

	256000	2.46	2.09	0.74	6.43	6.32	2.21	74	71	48	1374	1233	1087
TNP	2000	4.61	4.29	2.01	7.68	7.15	2.21	23	22	22	45	81	163
	4000	4.32	4.31	1.22	7.93	7.41	2.14	26	23	25	55	85	185
	8000	4.21	4.17	1.35	7.65	7.32	2.11	26	25	24	61	115	205
	16000	3.85	3.79	0.97	7.79	7.12	2.08	30	29	25	97	154	225
	32000	3.69	3.56	0.96	7.52	7.28	2.22	33	33	26	124	202	256
	64000	3.27	3.05	0.83	7.84	7.08	2.08	37	36	28	199	317	324
	128000	3.18	2.94	0.86	7.89	7.44	2.24	43	40	31	432	482	387
	256000	3.09	2.86	0.76	7.76	7.06	2.32	48	43	34	737	735	537
BIP	2000	7.42	7.32	2.04	8.46	8.31	2.71	85	81	69	1030	696	2237
	4000	7.33	7.04	1.43	8.35	8.44	2.64	85	83	70	1243	1208	2370
	8000	6.89	6.83	1.58	8.67	8.33	2.83	87	86	70	1497	1635	2623
	16000	6.67	6.26	0.94	8.51	8.42	2.48	91	90	72	1721	1888	2813
	32000	6.59	6.23	0.74	8.32	8.38	2.34	94	95	75	2882	3370	3484
	64000	6.42	5.87	0.81	8.68	8.39	2.71	99	100	82	4753	4066	4593
	128000	5.89	5.62	0.71	8.75	8.35	2.31	107	106	93	7811	6255	5567
	256000	5.82	5.29	0.68	8.48	8.34	2.19	117	116	102	10850	10410	6699

References

Ariyawansa, KA, Andrew J Felt. 2004. On a new collection of stochastic linear programming test problems. *INFORMS Journal on Computing* **16**(3) 291–299.

Asmussen, Søren, Peter W. Glynn. 2007. *Stochastic simulation: algorithms and analysis*, *Stochastic Modelling and Applied Probability*, vol. 57. Springer, New York.

Bayraksan, G., D.P. Morton. 2011. A sequential sampling procedure for stochastic programming. *Operations Research* **59**(4) 898–913.

Birge, J.R., F. Louveaux. 2011. *Introduction to stochastic programming*. Springer Verlag.

Bucklew, James Antonio. 2004. *Introduction to rare event simulation*. Springer Series in Statistics, Springer-Verlag, New York.

Dantzig, G.B., P.W. Glynn. 1990. Parallel processors for planning under uncertainty. *Annals of Operations Research* **22**(1) 1–21.

Devroye, Luc, László Györfi. 1985. *Nonparametric density estimation*. Wiley Series in Probability and Mathematical Statistics: Tracts on Probability and Statistics, John Wiley & Sons Inc., New York. The L_1 view.

Drew, S.S., T. Homem-de Mello. 2006. Quasi-monte carlo strategies for stochastic optimization. *Proceedings of the 38th conference on Winter simulation*. Winter Simulation Conference, 774–782.

Dupačová, Jitka, Nicole Gröwe-Kuska, Werner Römisch. 2003. Scenario reduction in stochastic programming. *Mathematical programming* **95**(3) 493–511.

Gelman, A., S. Brooks, G. Jones, X.L. Meng. 2010. *Handbook of Markov Chain Monte Carlo: Methods and Applications*. Chapman & Hall/CRC.

Haario, H., E. Saksman, J. Tamminen. 2001. An adaptive metropolis algorithm. *Bernoulli* 223–242.

Hall, P., S.N. Lahiri, Y.K. Truong. 1995. On bandwidth choice for density estimation with dependent data. *The Annals of Statistics* **23**(6) 2241–2263.

Higle, J.L. 1998. Variance reduction and objective function evaluation in stochastic linear programs. *INFORMS Journal on Computing* **10**(2) 236–247.

Higle, J.L., S. Sen. 1991. Stochastic decomposition: An algorithm for two-stage linear programs with recourse. *Mathematics of operations research* 650–669.

Homem-de Mello, T. 2006. On rates of convergence for stochastic optimization problems under non-iid sampling. *Manuscript, available on Optimization Online* .

Homem-de Mello, Tito, Vitor de Matos, Erlon Finardi. 2011. Sampling strategies and stopping criteria for stochastic dual dynamic programming: a case study in long-term hydrothermal scheduling. *Energy Systems* **2** 1–31. URL `http://dx.doi.org/10.1007/s12667-011-0024-y`. 10.1007/s12667-011-0024-y.

Infanger, G. 1992. Monte carlo (importance) sampling within a benders decomposition algorithm for stochastic linear programs. *Annals of Operations Research* **39**(1) 69–95.

Koivu, M. 2005. Variance reduction in sample approximations of stochastic programs. *Mathematical programming* **103**(3) 463–485.

Linderoth, J., A. Shapiro, S. Wright. 2006. The empirical behavior of sampling methods for stochastic programming. *Annals of Operations Research* **142**(1) 215–241.

Morton, D.P. 1998. Stopping rules for a class of sampling-based stochastic programming algorithms. *Operations research* 710–718.

Neddermeyer, Jan C. 2009. Computationally efficient nonparametric importance sampling. *Journal of the American Statistical Association* **104**(486) 788–802.

Parpas, Panos, Berç Rustem. 2007. Computational assessment of nested Benders and augmented Lagrangian decomposition for mean-variance multistage stochastic problems. *INFORMS J. Comput.* **19**(2) 239–247.

Pennanen, T., M. Koivu. 2005. Epi-convergent discretizations of stochastic programs via integration quadratures. *Numerische mathematik* **100**(1) 141–163.

Pereira, MVF, L. Pinto. 1991. Multi-stage stochastic optimization applied to energy planning. *Mathematical Programming* **52**(1) 359–375.

Powell, W.B. 2007. *Approximate Dynamic Programming: Solving the curses of dimensionality*, vol. 703. Wiley-Blackwell.

Ravindran, A. Ravi, ed. 2008. *Operations research and management science handbook*. Operations Research Series, CRC Press, Boca Raton, FL.

Roberts, Gareth O., Jeffrey S. Rosenthal. 2004. General state space Markov chains and MCMC algorithms. *Probab. Surv.* **1** 20–71. doi:10.1214/154957804100000024. URL `http://dx.doi.org/10.1214/154957804100000024`.

Rockafellar, R.T., R.J.B. Wets. 1991. Scenarios and policy aggregation in optimization under uncertainty. *Mathematics of operations research* 119–147.

Scott, David. 1992. *Multivariate Density Estimation: Theory, Practice, and Visualization (Wiley Series in Probability and Statistics)*. 1st ed. Wiley.

Shapiro, A. 2009. On a time consistency concept in risk averse multistage stochastic programming. *Operations Research Letters* **37**(3) 143–147.

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

Shapiro, A., D. Dentcheva, A.P. Ruszczyński. 2009. *Lectures on stochastic programming: modeling and theory*, vol. 9. Society for Industrial Mathematics.

Shapiro, A., T. Homem-de Mello. 1998. A simulation-based approach to two-stage stochastic programming with recourse. *Mathematical Programming* **81**(3) 301–325.

Silverman, Bernard. 1986. *Density Estimation for Statistics and Data Analysis (Chapman & Hall/CRC Monographs on Statistics & Applied Probability)*. 1st ed. Chapman and Hall/CRC.

Watson, J.P., R.J.B. Wets, D.L. Woodruff. 2010. Scalable heuristics for a class of chance-constrained stochastic programs. *INFORMS Journal on Computing* **22**(4) 543.

Zhang, Ping. 1996. Nonparametric importance sampling. *Journal of the American Statistical Association* **91**(435) 1245–1253.