INFORMS Journal on Computing
Vol. 27, No. 2, Spring 2015, pp. 358–377
ISSN 1091-9856 (print) | ISSN 1526-5528 (online)

http://dx.doi.org/10.1287/ijoc.2014.0630

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

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

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

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

Quang Kha Tran
Department of Computer Science, Imperial College London, South Kensington, United Kingdom 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. Our framework combines Markov chain Monte Carlo methods with kernel density estimation algorithms to build a nonparametric 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.

Keywords: Benders' decomposition; cutting plane algorithms; stochastic optimization; stochastic programming; importance sampling; variance reduction; Monte Carlo; Markov chain Monte Carlo; kernel density estimation; nonparametric

History: Accepted by Karen Aardal, Area Editor for Design and Analysis of Algorithms; received November 2013; revised May 2014; accepted September 2014.

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 multidimensional 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 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 is unknown and cannot be directly used in practice. However, it can be indirectly 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 based on the zero-variance distribution. The distribution was developed under the assumption that the uncertainty is modeled using discrete random variables. To the knowledge of the authors, it has not been extended to the continuous case. More importantly, the importance sampling distribution developed in their work would work well if the cost surface is approximately additively separable in the random dimensions. This is a difficult assumption to verify in practice and in our experience the approach in Dantzig and Glynn (1990) and Infanger (1992) can often perform much worse than the naive Monte Carlo method (see example in §4.6).

Our framework, which we refer to as the Markov chain Monte Carlo (MCMC) importance sampling (MCMC-IS) framework, exploits the fact that the zero-variance distribution is known up to a normalizing constant. This fact is well known and exploited by many other importance sampling algorithms. The first step is to use a Markov chain Monte Carlo algorithm to generate samples from the zero-variance distribution, and then 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; nonparametric, in that it does not require users to specify a family of distributions; robust, in that it consistently gives reasonable 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 that 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 Drew and Homem-de-Mello (2008). The non-independent and identically distributed (i.i.d.) case of MC sampling has been studied in Homem-de-Mello (2008). Control variates were proposed in Shapiro and Homem-de-Mello (1998) and 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, 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 multimodal 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, Bayraksan and Pierre-Louis 2012), chance-constrained programming (Watson et al. 2010, Barrera et al. 2014), and risk-averse stochastic programming (Shapiro 2009, Kozmık and Morton 2014).

Although 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 nonparametric importance sampling methods in the field of statistics (Zhang 1996, Neddermeyer 2009). However, these techniques

have not been adopted for practical applications because of the computational overhead involved in building a nonparametric importance distribution. In this paper, we demonstrate that the computational overhead of using an MCMC and KDE procedure is 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 run time.

Our paper is structured as follows: in §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 §3, we introduce the MCMC-IS framework and present readers with a set of theoretical insights and practical guidelines. In §4, we use a series of numerical experiments based on a simple newsvendor problem to illustrate the sampling-based properties of MCMC-IS and demonstrate the benefits of pairing MCMC-IS with a decomposition algorithm to solve stochastic programming models. In §5, we demonstrate that these benefits generalize across a collection of benchmark stochastic programming models. We summarize our contributions and discuss directions for future research in §6.

2. Motivation

We consider a multistage linear stochastic programming model defined as:

$$\begin{aligned} z^* = \min_{x_1} \quad & \{c_1^{\mathrm{T}} x_1 + \mathcal{Q}_1(x_1)\} \\ \text{s.t.} \quad & A_1 x_1 = b_1, \\ & x_1 \geq 0, \end{aligned} \tag{1}$$

where $c_1 \in \mathbb{R}^{n_1}$, $A_1 \in \mathbb{R}^{m_1 \times n_1}$, 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

$$\mathcal{Q}_t(x_t) = \mathbb{E}[Q_t(x_t, \xi_{t+1})], \quad t = 1, \ldots, T-1. \tag{2}$$

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:

$$\begin{aligned} Q_{t-1}(\hat{x}_{t-1}, \xi_t) = \min_{x_t} \quad & c_t^{\mathrm{T}}(\xi_t) x_t + \mathcal{Q}_t(x_t) \\ \text{s.t.} \quad & A_t(\xi_t) x_t = b_t(\xi_t) - W_t(\xi_t)\hat{x}_{t-1}, \\ & \qquad\qquad t = 2, \ldots, T \\ & x_t \geq 0, \end{aligned} \tag{3}$$

where $Q_T(\hat{x}_{T-1}, \xi_T) \equiv 0$ without loss of generality. We will assume that $c_t \in \mathbb{R}^{n_t}$, $A_t \in \mathbb{R}^{m_t \times n_t}$, $W_t \in \mathbb{R}^{m_t \times n_{t-1}}$, $b_t \in \mathbb{R}^{m_t \times 1}$. The components of these parameters are deterministic for $t = 1$, but may be random for $t = 2, \ldots, T$. 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 $\mathcal{Q} = \mathcal{Q}_1$, $W = W_2$, $\xi = \xi_2$, and $\hat{x} = \hat{x}_1$. 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. In this context, discretization means selecting a finite number of scenarios from a continuous distribution. If the problem is modeled with a discrete distribution, then discretization means selecting a smaller number of samples from the finite but often large number of realizations of the discrete distribution. Of course Monte Carlo sampling, including importance sampling, can be used to perform the discretization.

We refer to the discretization approach described previously as the "discretize-then-solve" approach and the resulting discretization as a scenario tree. An alternative to the discretize-then-solve approach is to generate different samples of the random parameters on the fly and use a 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 the following, 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

Benders-type 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 $\hat{x}_t$. The supporting hyperplanes are

also known as cuts. Given a fixed value $\hat{x}_t$, a cut takes the form of a linear inequality constraint

$$\mathcal{Q}_t(x_t) \geq \mathcal{Q}_t(\hat{x}_t) + \partial\mathcal{Q}_t(\hat{x}_t)(x_t - \hat{x}_t), \tag{4}$$

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 (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 a MC procedure in which we first generate a set of N samples of the random variables, $\xi_t^1, \ldots, \xi_t^N$, and then compute

$$\begin{aligned} \hat{\mathcal{Q}}_t^{\mathrm{MC}}(\hat{x}_t) &= \frac{1}{N}\sum_{i=1}^{N} Q_t(\hat{x}_t, \xi_t^i) \\ \widehat{\partial\mathcal{Q}}_t^{\mathrm{MC}}(\hat{x}_t) &= \frac{1}{N}\sum_{i=1}^{N} \partial Q_t(\hat{x}_t, \xi_t^i). \end{aligned} \tag{5}$$

Although MC methods can significantly reduce the computational burden in generating cuts relative to the discretize-then-solve 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 well-known phenomenon in Figure 1, where we plot the sampled cuts that are produced when a CMC method is paired with a decomposition algorithm to solve a simple two-stage newsvendor problem, whose parameters are specified in §4.1.

Both cuts in this example were constructed using $N = 50$ samples. For clarity, we plot a subset of the sample values $Q(\hat{x}, \xi_i)$, $i = 1, \ldots, N$ along the vertical line of $\hat{x}$, 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

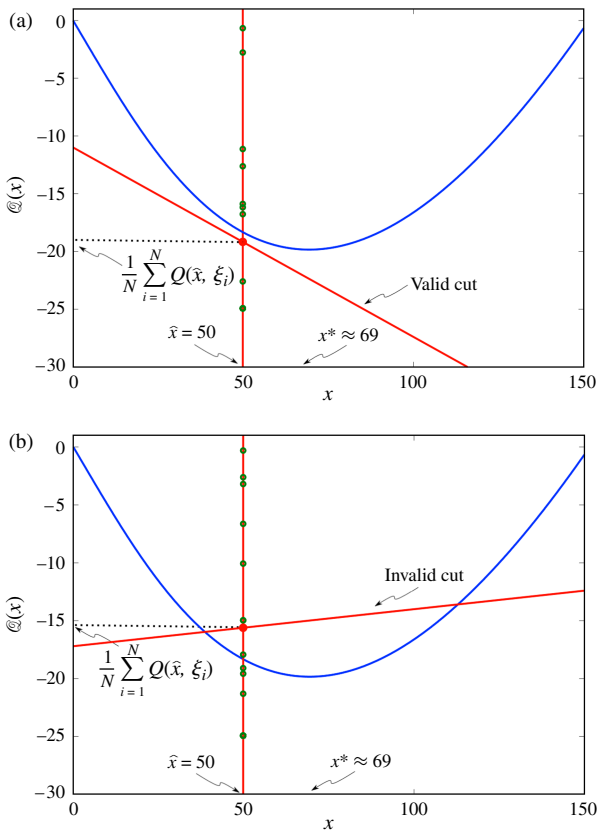

Figure 1 **(Color online) In 1(a) the Sampled Cut Is Valid; Assuming That Only Valid Cuts Are Generated in Subsequent Iterations, a Decomposition Algorithm Will Produce Accurate Estimates of x^* and z^*. In 1(b) the Sampled Cut Is Invalid; Even If All The Other Cuts Produced by the Algorithm Are Valid, the True Optimal Solution at x^* Will Remain Infeasible, and a Decomposition Algorithm Will Produce High-Error Estimates for the Optimal Value and Solution.**

using a scenario tree. Although this approach allows us to calculate the exact values of the parameters in (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 (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. One approach is to use a very large scenario tree or a continuous distribution and then use scenario reduction methods to find a representation with a finite and manageable scenario tree that is close to the original in some sense (see, e.g., 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 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 $\hat{\mathcal{Q}}^{\mathrm{MC}}$ in (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 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

$$\mathcal{Q}(\hat{x}) = \mathbb{E}_f[Q(\hat{x},\xi)] = \mathbb{E}_g[Q(\hat{x},\xi)\Lambda(\xi)], \tag{6}$$

where $\mathbb{E}_f$ and $\mathbb{E}_g$ denote expectation with the distributions f and g, respectively. The function $\Lambda\colon \Xi \to \mathbb{R}$ is called the likelihood function and it is given by

$$\Lambda(\xi) = \frac{f(\xi)}{g(\xi)}. \tag{7}$$

The likelihood function 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

$$\hat{\mathcal{Q}}^{\mathrm{IS}}(\hat{x}) = \frac{1}{N}\sum_{i=1}^{N} Q(\hat{x},\xi_i)\Lambda(\xi_i). \tag{8}$$

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 $\hat{x}$. It is easy to show that the variance of an importance sampling estimate is minimized when we sample from

$$g^*(\xi) = \frac{|Q(\hat{x},\xi)|}{\mathbb{E}_f|Q(\hat{x},\xi)|} f(\xi). \tag{9}$$

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 (9) 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 (9) to construct zero-variance estimates if and only if we already have a zero-variance estimate of $\mathbb{E}_f|Q(\hat{x},\xi)|$.

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 (9) using a 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 (7). In this case, the likelihood of a given sample is given by

$$\Lambda^*(\xi) = \frac{\mathbb{E}_f|Q(x,\xi)|}{|Q(x,\xi)|}, \tag{10}$$

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: although we cannot use the samples from (9) 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, 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 a 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 (9) 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 to generate a Markov chain that has (9) as its stationary distribution. In the kth step, the algorithm generates a proposed state ζ_k using a proposal distribution whose density $q(\cdot\,|\,\xi_k)$ 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 k of the Metropolis-Hastings algorithm is sampled from $q(\cdot\,|\,\xi_k)$ as

$$\zeta_k=\xi_k+v_k, \tag{11}$$

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 follows:

$$a(\xi_k,\zeta_k)=\min\left\{1,\frac{|Q(\hat{x},\zeta_k)|f(\zeta_k)}{|Q(\hat{x},\xi_k)|f(\xi_k)}\right\}. \tag{12}$$

Once a set of M samples has been generated from the zero-variance distribution specified in (9) using a MCMC algorithm, we can construct an approximate zero-variance distribution from these samples using a KDE algorithm. KDE algorithms are established techniques 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

$$\hat{g}_M(\xi)=\frac{1}{M}\sum_{i=1}^{M}K_H(\xi,\xi_i), \tag{13}$$

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. To ensure that $\hat{g}$ is a proper probability density function, we impose the following conditions on the kernel function:

$$\begin{aligned} &K_H(\cdot\,,\cdot)\geq 0,\\ &\int_{\Xi}K_H(\xi,\cdot)d\xi=1. \end{aligned} \tag{14}$$

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:

$$K_H(\xi,\xi_i)=\prod_{k=1}^{D}\frac{1}{\sqrt{2\pi}h_k}\exp\left(\frac{(\xi_k-\xi_{i,k})^2}{2h_k^2}\right). \tag{15}$$

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 $\hat{g}_M$, we can finally construct an importance sampling estimate of the recourse function by generating N additional samples from $\hat{g}_M$. 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 $\hat{g}_M$ that is similar to g^*. Generating samples from $\hat{g}_M$ is also beneficial in that the samples are independent and the kernel functions are easy to sample from. In practice, we construct $\hat{g}_M$ using modest values of M and then construct $\hat{\mathcal{Q}}^{\mathrm{IS}}(\hat{x})$ 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, with additional implementation details in §3.4.

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

Require: $\hat{x}$: previous stage decision
Require: M: number of samples generated using the MCMC algorithm
Require: N: number of samples generated with the approx. zero-variance distribution
Require: ξ_0: starting state of the MCMC algorithm

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

1.1 Set $k = 0$
1.2 Given the current state ξ_k, generate $\zeta_k \sim q(\cdot \mid \xi_k)$.
1.3 Generate a uniform random variable $u \sim U \in (0, 1)$.
1.4 Transition to the next state according to:

$$\xi_{k+1} = \begin{cases} \zeta_k & \text{if } u \le a(\xi_k, \zeta_k), \\ \xi_k & \text{otherwise,} \end{cases}$$

where

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

1.5 Let $k \leftarrow k+1$. If $k = M$ then proceed to Step 2. Otherwise return to Step **1.1**.

`Step 2: Construct the Zero-Variance Distribution using KDE`

2.1 Reconstruct the approximate zero-variance distribution:

$$\hat{g}_M(\xi) = \frac{1}{M}\sum_{i=1}^{M} K_H(\xi, \xi_i).$$

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

3.1 For Generate N new samples from $\hat{g}_M$ and form the importance sampling estimate

$$\hat{\mathcal{Q}}^{\mathrm{IS}}(\hat{x}) = \frac{1}{N}\sum_{i=1}^{N} Q(\hat{x}, \xi_i)\frac{f(\xi_i)}{\hat{g}_M(\xi_i)}.$$

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

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 (where h_k denotes the kth diagonal entry in the H matrix). In particular, if $(Mh^D)^{-1} \to \infty$, $h \to 0$ as $M \to \infty$, and the density function is approximated as

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

then it has been shown that $\hat{g}_M$ 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—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 a MCMC algorithm (M). This is an important choice because 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 $M = 3{,}000$ within all of our numerical experiments in §§4 and 5. It may be surprising that a small and constant number is sufficient even for large-scale problems. In §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 toward 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 before implementing MCMC-IS is the KDE algorithm used in the construction of the approximate 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 that 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 and leave-one-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 §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 §§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 §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 §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. Further experimental results on MCMC-IS can be found in Ustun (2012).

4.1. Description of the Newsvendor Problem

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

$$\begin{aligned} z^* = \min_x \ & \{x + \mathcal{Q}(x)\} \\ \text{s.t.} \ & x \geq 0, \end{aligned} \tag{16}$$

and the recourse function is a linear program defined as

$$\begin{aligned} Q(\hat{x}, \xi) = \min_{y_1, y_2} & \{-p(\xi) y_1 - r y_2\} \\ y_1 &\leq d(\xi), \\ y_1 + y_2 &\leq \hat{x}, \\ y_1, y_2 &\geq 0, \end{aligned} \tag{17}$$

where $\hat{x}$ 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 independent normal random variables with mean μ 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 because 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 $\mathrm{MSE}(\tilde{x}) \equiv \mathbb{E}\,\|x^* - \tilde{x}\|_2^2$.
- Mean-squared error of the estimate of the optimal value $\tilde{z}$, defined as $\mathrm{MSE}(\tilde{z}) \equiv \mathbb{E}\,\|z^* - \tilde{z}\|_2^2$.
- Mean-squared error of the approximate zero-variance distribution, defined as $\mathrm{MSE}(\hat{g}) \equiv \int (g(\xi) - \hat{g}(\xi))^2 d\xi$.
- Mean-squared error of the estimated recourse function $\hat{\mathcal{Q}}$ at a fixed point $\hat{x}$, defined as $\mathrm{MSE}(\hat{\mathcal{Q}}(\hat{x})) \equiv \mathbb{E}\,\|\mathcal{Q}(x) - \hat{\mathcal{Q}}(\hat{x})\|_2^2$.
- Sample variance of the estimated recourse function $\hat{\mathcal{Q}}$ at a fixed point $\hat{x}$, defined as $\mathrm{S}(\hat{\mathcal{Q}}(\hat{x}))^2 \equiv \mathbb{E}[\mathcal{Q}(x) - \mathbb{E}[\hat{\mathcal{Q}}(\hat{x})]]^2$.

In our experiments we estimated the quantities above using a sample average approximation. 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 we 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 a MCMC-IS uses a total of $M + M_r$ functional evaluations to construct an importance sampling distribution, where M_r denotes the number of samples that are rejected because of 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 version 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 §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 available at 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 Metropolis-Hastings algorithm, and a MEX file that can generate samples from 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 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 ($\hat{g}_M$) 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 $\hat{g}_M$. 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

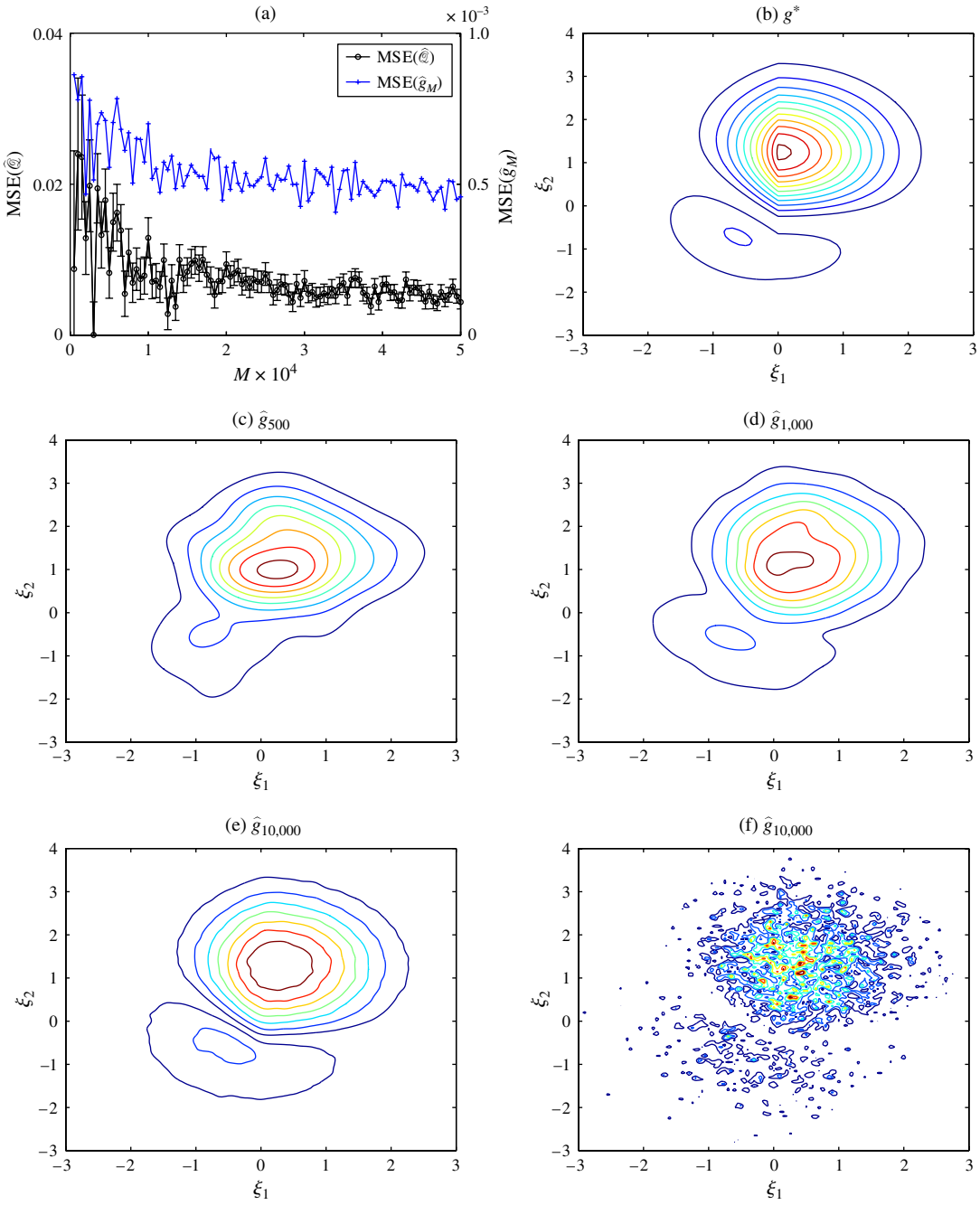

Figure 2 **(Color online) (a) The Majority of the Gains in Variance Reduction and Accuracy Can Be Achieved for Small Values of M. Note that the Axis for MSE($\hat{g}_M$) Is on the Right, and the Scale for MSE($\hat{Q}$) Is on the Left. (b) Contours of g^*. (c)–(e) Contours of $\hat{g}_M$ for Different Values of M; the Bandwidth Parameter for These Distributions Is Estimated Using a One-Dimensional Likelihood-Based Search. (f) $\hat{g}_{10,000}$ 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)**

for this empirical observation is that if our $\hat{g}_M$ qualitatively agrees with g^*, then the sample statistics of the approximate distribution will qualitatively agree with the sample statistics of the zero-variance density.

To illustrate this point, we plot the contours of the true zero-variance distribution g^* in Figure 2(b) and the contours of $\hat{g}_M$ for different values of M in Figures 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 $\hat{g}_M$ by approximately 12% but increases its variance by approximately 15%, thereby demonstrating

the bias-variance trade-off of KDE algorithms. These results were constructed with the first-stage decision fixed at $\hat{x} = 50$.

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 $\hat{g}_M$ as opposed to the original distribution f. As a result, the samples that are generated using $\hat{g}_M$ 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) whereas 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 $\hat{g}_M$ using MCMC-IS. The first set of contours in Figure 3 pertains to the original distribution f and 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 $\hat{x}_r = 50$ and two other approximate zero-variance distributions constructed around the point

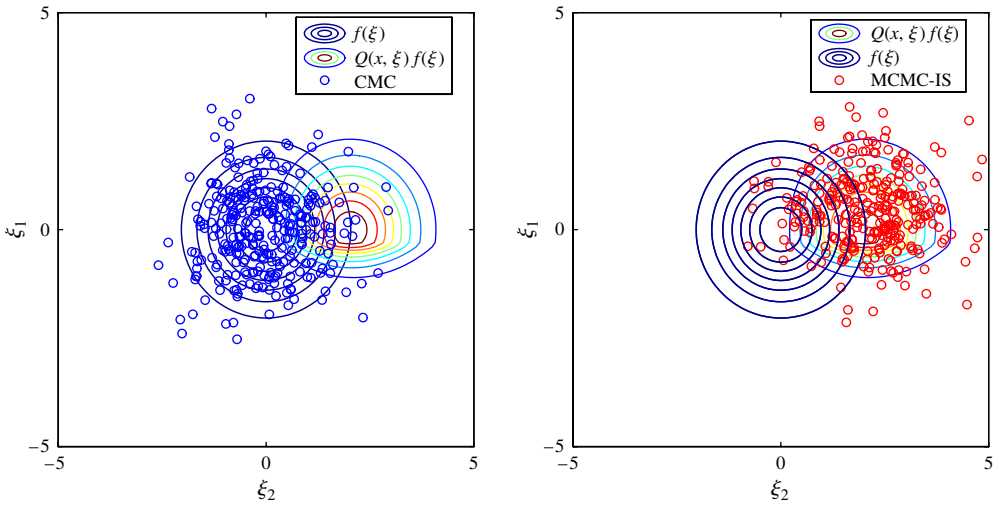

Figure 3 **(Color online) Comparison of Points Generated with the Standard CMC Approach, and MCMC-IS**

Note. Left: Using MC sampling; Right: Using importance sampling.

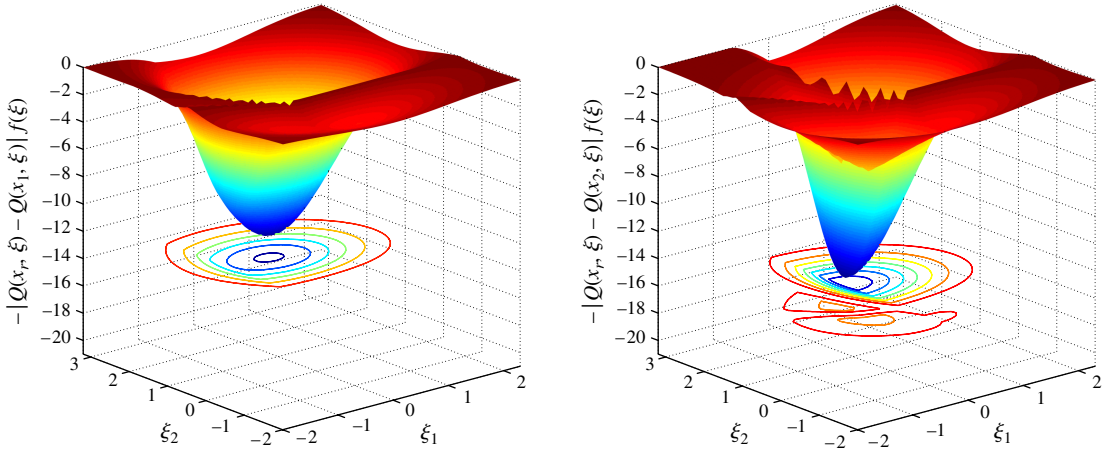

Figure 4 **(Color online) The Absolute Difference Between an Approximate Zero-Variance Distribution Constructed at $\hat{x}_r = 50$ and Two Other Approximate Zero-Variance Distributions Constructed at $\hat{x}_1 = 10$ (Left) and $\hat{x}_2 = 100$ (Right)**

$\hat{x}_1 = 10$ (left) and $\hat{x}_2 = 100$ (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.

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 a 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 §4.1. In this regime, MCMC-IS outperforms all other methods (Figures 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

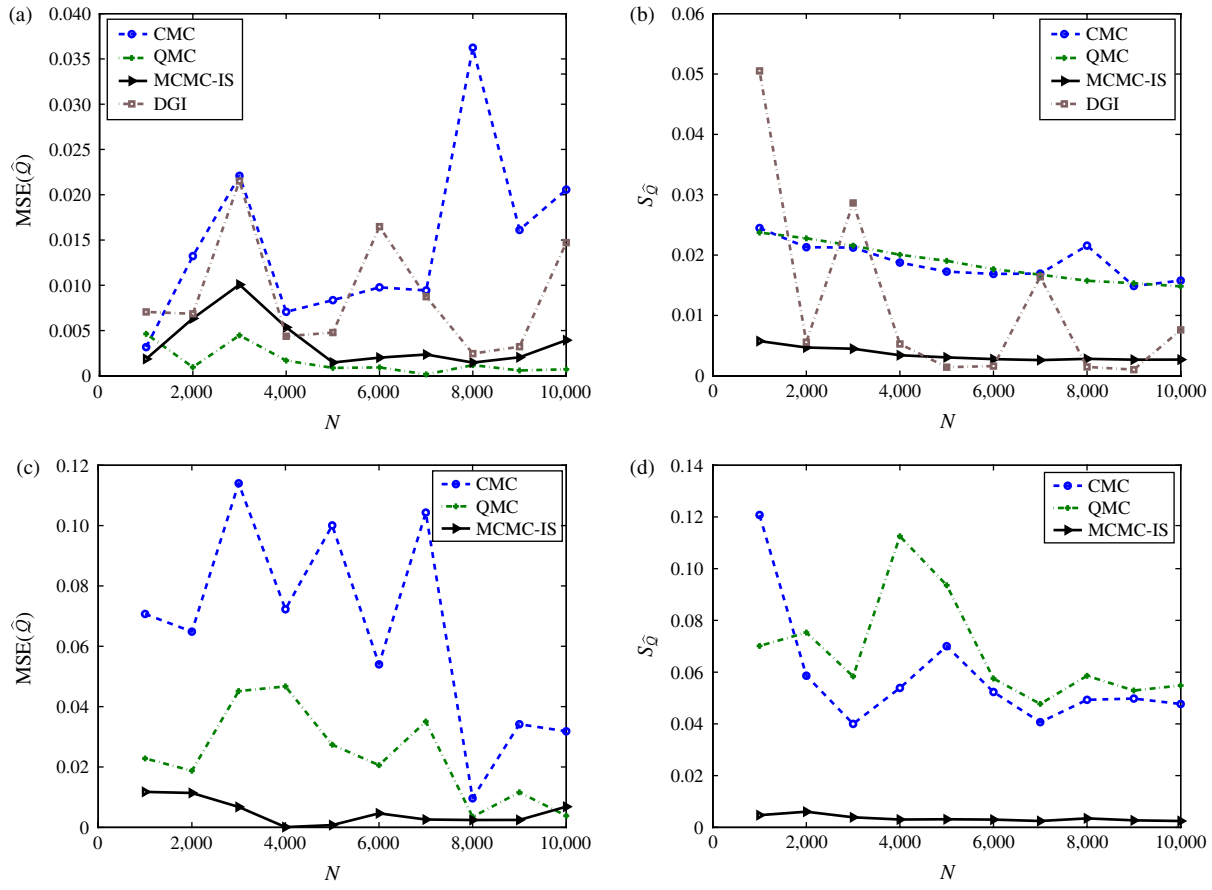

Figure 5 **(Color online) (a) and (b): Comparison of the Accuracy and Variance of Estimates Produced by Different Methods for a Moderate-Variance Problem with $\sigma = 1$. (c) and (d): Comparison of the Accuracy and Variance of Estimates Produced by Different Methods for a Higher-Variance Problem with $\sigma = 2$**

Notes. Note that we omit the results for the DGI method when $\sigma = 2$ for clarity. The normalized values of $S_{\hat{Q}}$ and MSE($\hat{Q}$) for DGI are around 20% and 40%, respectively.

that involve rare events (Ravindran 2008). Unfortunately, such complex probability distributions are difficult to include in stochastic programming models, because existing variance reduction methods will need an extremely large number of samples 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 for two reasons. First, as was illustrated in §3, 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 must also be able to accurately evaluate the likelihood of each sample that it generates as in (7) 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.

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 Figure 6(a). In this example, we have replaced the original integrand in the recourse function $Q(\hat{x}, \xi_1, \xi_2) f(\xi_1, \xi_2)$ with a new integrand $Q(\hat{x}, w(\xi_1), w(\xi_2)) f(\xi_1, \xi_2)$, in which $w(\xi) = \exp(\xi^2/2 - (\xi+3)^2/8) + \exp(\xi^2/2 - (\xi+1)^2/8)$, $\hat{x} = 50$, 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.

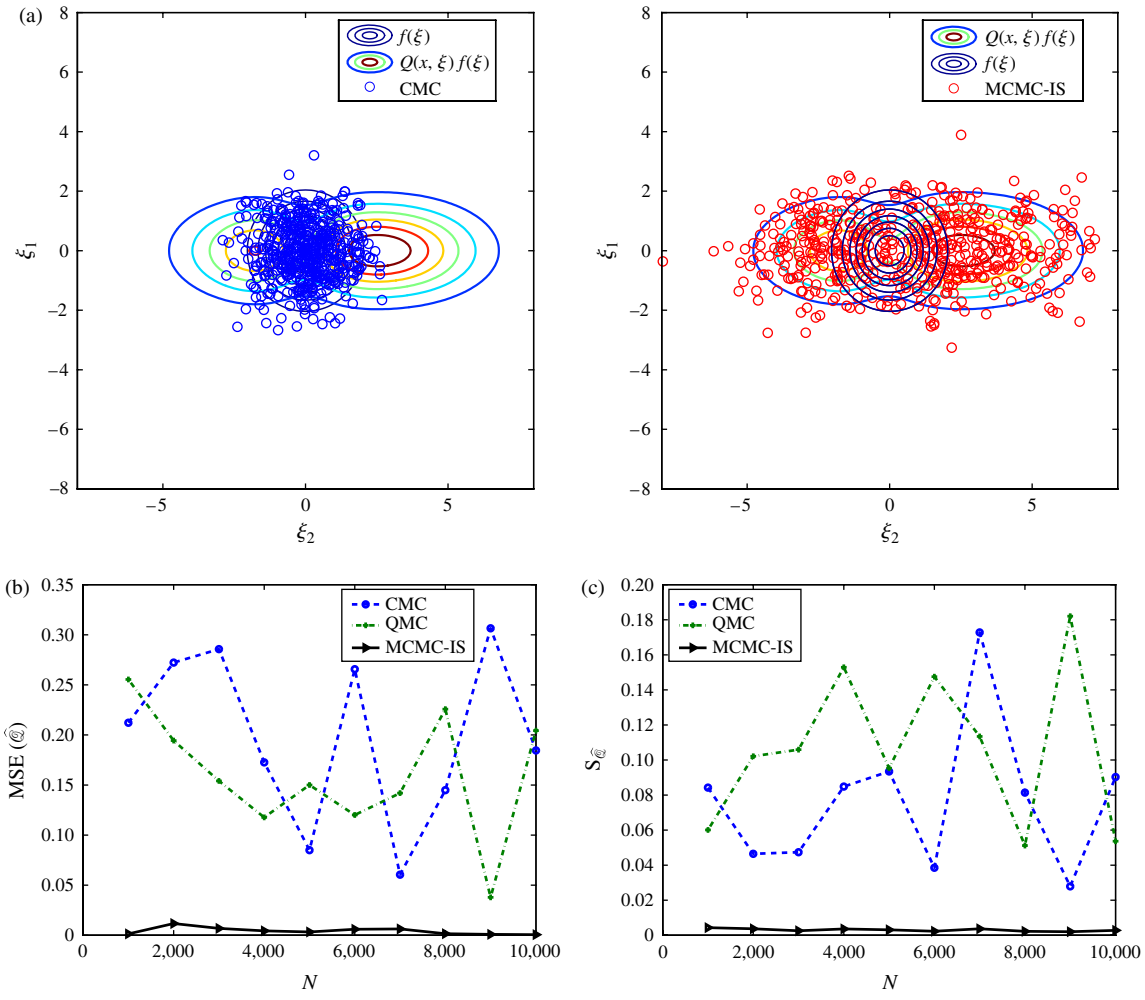

Figure 6 (Color online) (a): Contours of a Multimodal Model. Samples Generated Using CMC Are Shown on the Left and the Samples from MCMC-IS Are Shown on the Right. (b) and (c): Error and Variance of Estimates Produced by Different Methods

As in §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 panel). 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 distribution $\hat{g}_M$ 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 (Higle 1998, §1.4).

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 that has been paired with MCMC-IS, CMC, and QMC.

We consider an extension of the newsvendor problem from §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 §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 §§4.3–4.7, 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 log-normal distributions from §4.1, and we build the approximate zero-variance distribution for each sampled cut using $M = 3{,}000$ 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 §4.7, and we build the approximate zero-variance distribution for each sampled cut using $M = 3{,}000$ 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 to solve a collection of benchmark stochastic programming models. Initial numerical results appeared in Ustun (2012); we report results on a larger set of test problems next.

5.1. Overview of the Test Problems

To verify that our findings from §4.8 generalize to stochastic programming models, we have based the

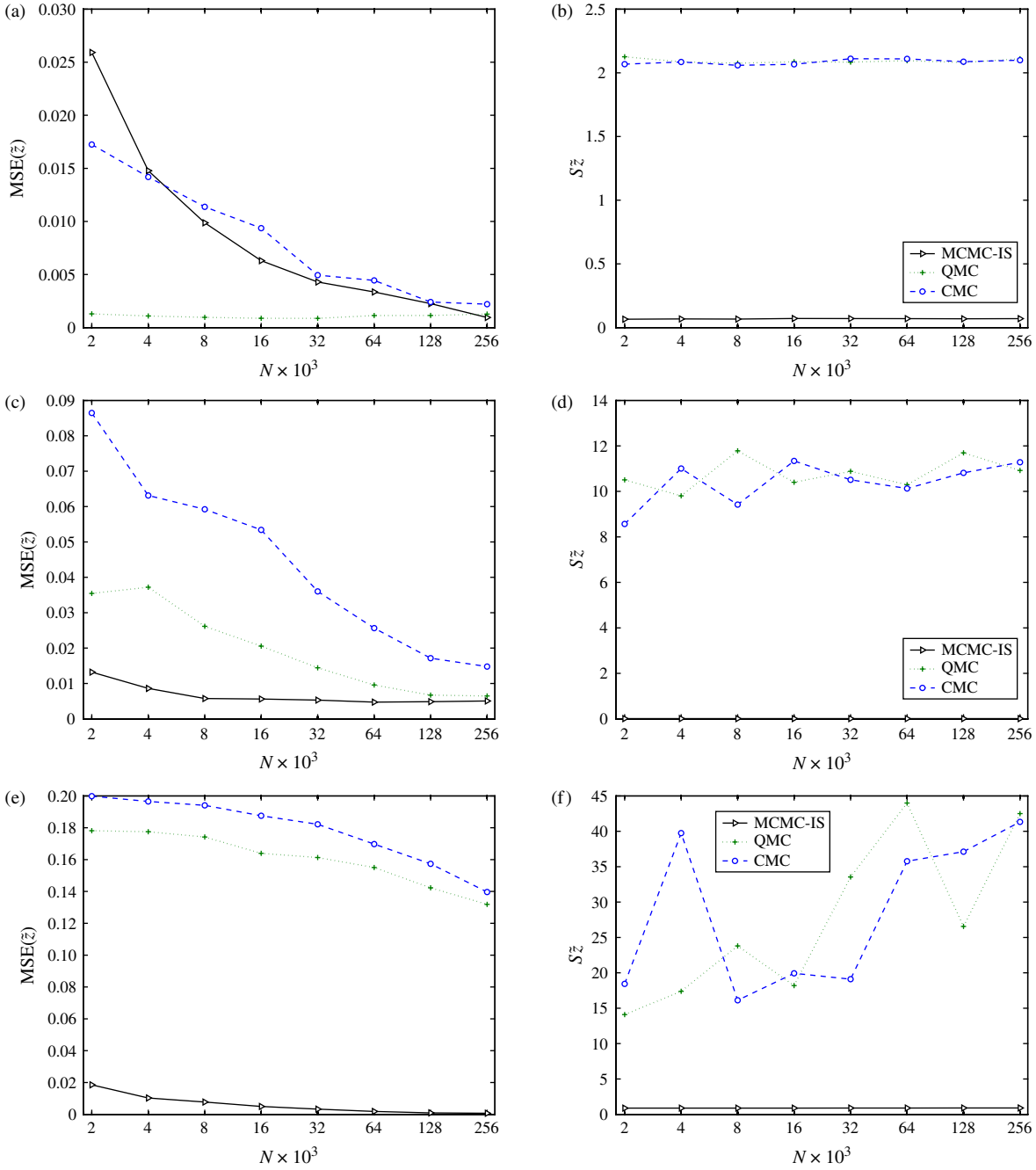

Figure 7 **(Color online) Error and Variance of Estimates for a Newsvendor Problem Where the Uncertainty in Demand and Sales Price Is Modeled Using a Lower-Variance Lognormal Distribution with $\sigma = 1$ (7(a)–7(b)), a Higher-Variance Lognormal Distribution with $\sigma = 2$ (7(c)–7(d)), and Multimodal Rare-Event Distribution (7(e)–7(f))**

numerical experiments in this section on a collection of nine benchmark stochastic programming models from Ariyawansa and Felt (2004). Table 1 shows the number of stages and number of random variables for the test problems used in the numerical experiments. We have specifically chosen these models because 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.

It is worth noting that many of the problems in Ariyawansa and Felt (2004) had to be modified to be solved with a sampling-based approach. This was because many problems were originally formulated

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

Problem	No. of stages (T)	No. of random variables ($\sum_t D_t$)
Airlift operation scheduling (AOS)	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

using discrete distributions and scenario trees (sometimes with three 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 because it required the use of a nonlinear programming solver. The full details of our modifications are listed in the online supplement, Section A (available as supplemental material at http://dx.doi.org/10.1287/ijoc.2014.0630).

5.2. Details on the Numerical Experiments

As in §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 $M = 3{,}000$ 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 $N = 2{,}000$ to $N = 256{,}000$. 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. Table 4 in the online supplement, Section B gives the average number of rejected samples from MCMC.

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 that achieves a value that is within 10% of the optimal value with 95% confidence.

To report error statistics as in §4, 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.

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 show the median 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 nine 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 section B of the online supplement. 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 $N = 256{,}000$ samples; the other two sampling methods exhibit an extremely slow convergence to the optimal value and require a far greater number of samples to converge.

In Figure 8(d) we plot the error times CPU time for each method (in % error × 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

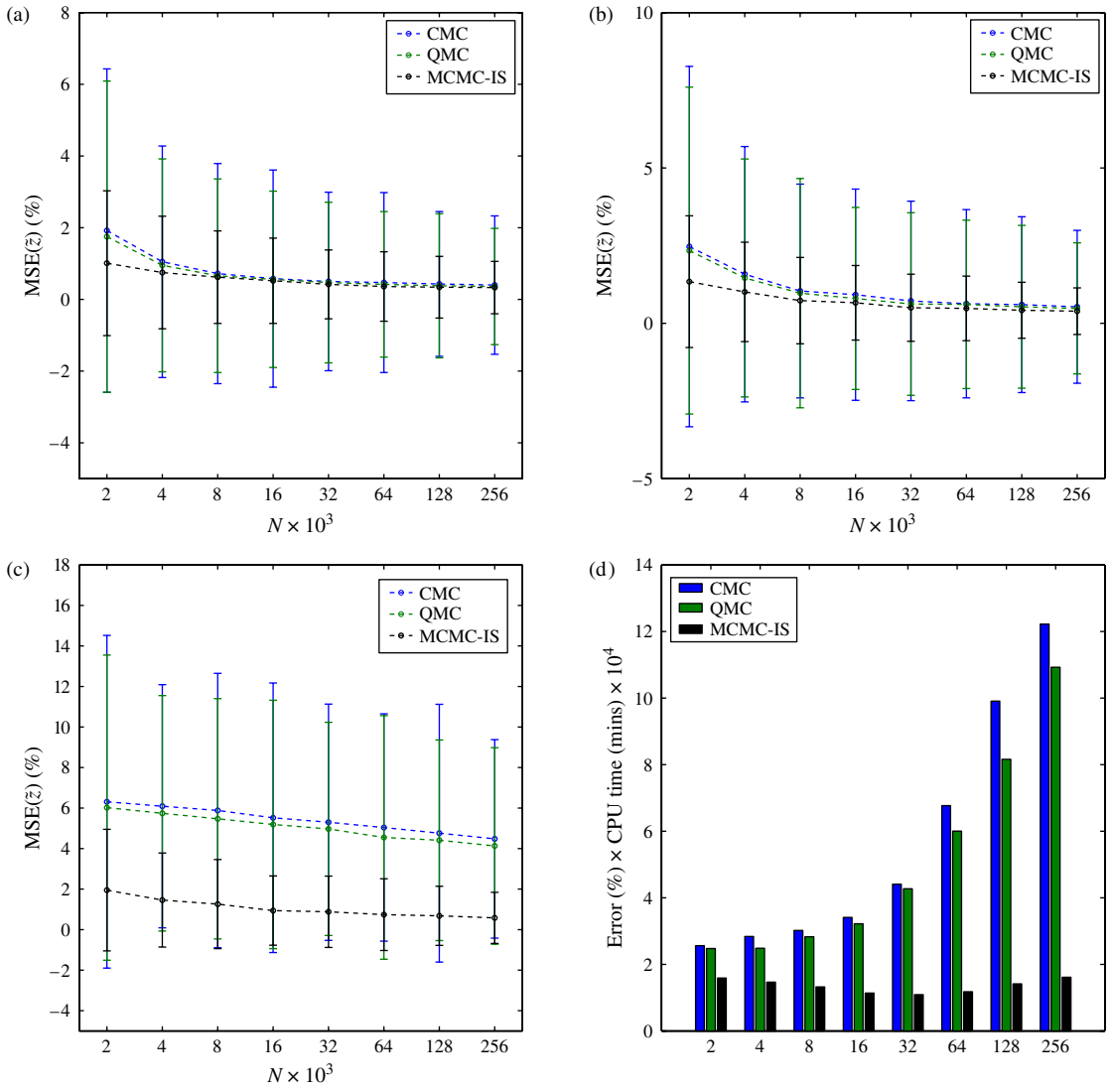

Figure 8 (Color online) Median Results with the Collection of Test Problems

Notes. (a) MSE($\bar{z}$) for models with lower-variance distributions. (b) MSE($\bar{z}$) for models with higher-variance distributions. (c) MSE($\bar{z}$) for models with rare-event distribution. The error bars indicate the standard error associated with the solution obtained. (d) Error (%) × CPU Time (mins); for this plot we averaged the low variance, moderate variance, and rare event 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., $N = 2{,}000$), 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., $N = 8{,}000$ and onward), we see that the cost of MCMC-IS relative to other methods (while taking into consideration the error reduction) is much less.

5.4. When to Use MCMC-IS in Stochastic Programming

The proposed algorithm has an additional overhead when compared to CMC and QMC. The benefits of variance reduction are obvious from Figures 8(a)–8(c). In Figure 8(d) we showed that the algorithm is also efficient in the sense defined in §4. Depending on the application, the efficiency measure we used may or may not be appropriate. We therefore conclude the discussion on our numerical results by weighing up the CPU overhead and standard error statistics from our experiments. Based on these statistics, we offer some insights on when the proposed algorithm is expected to outperform conventional sampling methods.

In Table 2, columns two to four present the computational overhead of MCMC-IS when compared to either CMC or QMC (who chose the best from the two). We report the median computational overhead across the different problems in percentage terms and in parenthesis we tabulate the median overhead

Table 2 Computational Overhead and Variance Reduction Trade-offs for MCMC-IS

	Median overhead (%) (secs)			Median var. reduction (%)		
No. of samples ($N \times 10^3$)	$\sigma = 1$	$\sigma = 2$	Rare	$\sigma = 1$	$\sigma = 2$	Rare
2	111% (40)	134% (80)	61% (103)	45	56	74
4	92% (40)	143% (96)	80% (154)	50	57	73
8	79% (43)	118% (90)	61% (144)	48	54	75
16	74% (46)	71% (78)	47% (133)	48	58	74
32	58% (51)	25% (37)	16% (61)	47	56	73
64	11% (16)	−20% (−61)	−8% (−45)	50	56	72
128	−9% (−20)	−29% (−141)	−15% (−126)	53	62	73
256	−20% (−60)	−30% (−208)	−23% (−285)	50	62	74

in seconds. At first glance it may seem that SDDP combined with MCMC-IS does not become competitive until the number of samples becomes large (around $N = 64 \times 10^3$). However, CPU time alone is not sufficient to judge the performance of sampling algorithms. Accuracy is also an important consideration. To illustrate the trade-offs, consider the results for $\sigma = 1$ for which our algorithm appears to be the least competitive. In this low variance regime we still manage to have half the standard error of CMC/QMC even for very large N. It is well known that to halve the standard error of Monte Carlo estimates, one needs to increase the number of samples by four. As a result our algorithm becomes competitive not around $N = 64 \times 10^3$ but around $N = 16 \times 10^3$ to achieve a comparable level of accuracy. Whether this value is too large to justify MCMC-IS will depend on the application. Many engineering applications, especially in energy systems, require a large number of samples to obtain a sufficiently accurate approximation of the recourse function. For example, in Lubin et al. (2011) the authors found that they need $10^4 - 10^5$ scenarios to represent a realistic energy system with uncertainties distributed across time and space. Similarly, models in finance can also require a large number of scenarios (Gondzio and Grothey 2006). When the problem has higher variance, the number of samples for which our algorithms becomes competitive is even lower. Finally, when the model has rare events, one would expect that a large number of samples should be used to accurately capture the uncertainties. Given the large error of other methods (see Figure 8(c)), the proposed algorithm clearly outperforms the other methods in the sense that it has about 70% less variance and in terms of CPU time becomes competitive after a moderate number of samples. In summary, we believe that the proposed method should be used when the model has a moderate/high variance or rare events, and when more than 16×10^3 samples are required to estimate the recourse function.

6. Conclusions

Multistage stochastic programming models are considered to be 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 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 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 has a 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.

Supplemental Material

Supplemental material to this paper is available at http://dx.doi.org/10.1287/ijoc.2014.0630.

Acknowledgments

The authors thank the area editor and two anonymous referees for their helpful comments that led to substantial improvements of the paper. The work of the first author was partially supported by a FP7 Marie Curie Career Integration [Grant PCIG11-GA-2012-321698 SOC-MP-ES] and Engineering and Physical Sciences Research Council (EPSRC) [Grant EP/K040723/1]. The work of the third author was partially supported by U.S. National Science Foundation [Grant No. 835414], and by the U.S. Department of Energy, Office of Science, Biological and Environmental Research Program, Integrated Assessment Research Program [Grant No. DE-SC0005171].

References

Ariyawansa KA, Felt AJ (2004) On a new collection of stochastic linear programming test problems. *INFORMS J. Comput.* 16(3): 291–299.

Asmussen S, Glynn PW (2007) *Stochastic Simulation: Algorithms and Analysis*, Stochastic Simulation: Algorithms and Analysis, Stochastic Modelling and Applied Probability, Vol. 57 (Springer-Verlag, New York).

Barrera J, Homem-de-Mello T, Moreno E, Pagnoncelli BK, Canessa G (2014) Chance-constrained problems and rare events: An importance sampling approach. Accessed April 10, 2015, http://www.optimization-online.org/DB_FILE/2014/02/4250.pdf.

Bayraksan G, Morton DP (2011) A sequential sampling procedure for stochastic programming. *Oper. Res.* 59(4):898–913.

Bayraksan G, Pierre-Louis P (2012) Fixed-width sequential stopping rules for a class of stochastic programs. *SIAM J. Optim.* 22(4):1518–1548.

Birge JR, Louveaux F (2011) *Introduction to Stochastic Programming* (Springer, New York).

Bucklew JA (2004) *Introduction to Rare Event Simulation*, Springer Series in Statistics (Springer-Verlag, New York).

Dantzig GB, Glynn PW (1990) Parallel processors for planning under uncertainty. *Ann. Oper. Res.* 22(1):1–21.

Devroye L, Györfi L (1985) *Nonparametric Density Estimation: The L_1 View*, Wiley Series in Probability and Mathematical Statistics (John Wiley & Sons, New York).

Drew SS, Homem-de-Mello T (2008) Quasi-Monte Carlo strategies for stochastic optimization. Perrone LF, Lawson BF, Liu J, Wieland FP, eds. *Proc. 38th Conf. Winter Simulation, Monterey, CA*, 774–782.

Dupačová J, Gröwe-Kuska N, Römisch W (2003) Scenario reduction in stochastic programming. *Math. Programming* 95(3):493–511.

Gelman A, Brooks S, Jones G, Meng XL (2010) *Handbook of Markov Chain Monte Carlo: Methods and Applications* (Chapman & Hall/CRC, Boca Raton, FL).

Gondzio J, Grothey A (2006) Direct solution of linear systems of size 10^9 arising in optimization with interior point methods. Wyrzykowski R, Dongarra J, Meyer N, Waśniewski J, eds. *Parallel Processing and Applied Mathematics*, LNCS 3911 (Springer, Berlin), 513–525.

Haario H, Saksman E, Tamminen J (2001) An adaptive Metropolis algorithm. *Bernoulli* 7(2):223–242.

Hall P, Lahiri SN, Truong YK (1995) On bandwidth choice for density estimation with dependent data. *Ann. Statist.* 23(6): 2241–2263.

Higle JL (1998) Variance reduction and objective function evaluation in stochastic linear programs. *INFORMS J. Comput.* 10(2): 236–247.

Higle JL, Sen S (1991) Stochastic decomposition: An algorithm for two-stage linear programs with recourse. *Math. Oper. Res.* 16(3):650–669.

Homem-de-Mello T (2008) On rates of convergence for stochastic optimization problems under non-independent and identically distributed sampling. *SIAM J. Optim.* 19(2):524–551.

Homem-de-Mello T, de Matos V, Finardi E (2011) Sampling strategies and stopping sriteria for stochastic dual dynamic programming: A case study in long-term hydrothermal scheduling. *Energy Systems* 2(1):1–31.

Infanger G (1992) Monte Carlo (importance) sampling within a Benders decomposition algorithm for stochastic linear programs. *Ann. Oper. Res.* 39(1):69–95.

Koivu M (2005) Variance reduction in sample approximations of stochastic programs. *Math. Programming* 103(3):463–485.

Kozmık V, Morton DP (2014) Evaluating policies in risk-averse multistage stochastic programming. *Math. Programming*, ePub ahead of print May 13, http://www.link.springer.com/article/10.1007/s10107-014-0787-8.

Linderoth J, Shapiro A, Wright S (2006) The empirical behavior of sampling methods for stochastic programming. *Ann. Oper. Res.* 142(1):215–241.

Lubin M, Petra CG, Anitescu M, Zavala V (2011) Scalable stochastic optimization of complex energy systems. Lathrop S, ed. *High Performance Comput., Networking, Storage Anal. (SC), 2011 Internat. Conf.* (IEEE, Piscataway, NJ), 1–10.

Morton DP (1998) Stopping rules for a class of sampling-based stochastic programming algorithms. *Oper. Res.* 46(5):710–718.

Neddermeyer JC (2009) Computationally efficient nonparametric importance sampling. *J. Amer. Statist. Assoc.* 104(486): 788–802.

Parpas P, Rustem B (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, Koivu M (2005) Epi-convergent discretizations of stochastic programs via integration quadratures. *Numerische Mathematik* 100(1):141–163.

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

Powell WB (2007) *Approximate Dynamic Programming: Solving the Curses of Dimensionality*, Vol. 703 (Wiley-Blackwell, Hoboken, NJ).

Ravindran AR, ed. (2008) *Operations Research and Management Science Handbook*, Operations Research Series (CRC Press, Boca Raton, FL).

Roberts GO, Rosenthal JS (2004) General state space Markov chains and MCMC algorithms. *Probab. Surv.* 1:20–71.

Rockafellar RT, Wets RJ-B (1991) Scenarios and policy aggregation in optimization under uncertainty. *Math. Oper. Res.* 16(1): 119–147.

Scott D (1992) *Multivariate Density Estimation: Theory, Practice, and Visualization*, 1st ed. (John Wiley & Sons, Hoboken, NJ).

Shapiro A (2009) On a time consistency concept in risk averse multistage stochastic programming. *Oper. Res. Lett.* 37(3): 143–147.

Shapiro A (2011) Analysis of stochastic dual dynamic programming method. *Eur. J. Oper. Res.* 209(1):63–72.

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

Shapiro A, Dentcheva D, Ruszczyński AP (2009) *Lectures on Stochastic Programming: Modeling and Theory*, Vol. 9 (Society for Industrial and Applied Mathematics, Philadelphia).

Silverman BW (1986) *Density Estimation for Statistics and Data Analysis*, 1st ed. (Chapman and Hall, Boca Raton, FL).

Ustun B (2012) The Markov Chain Monte Carlo approach to importance sampling in stochastic programming. Master's thesis, Massachusetts Institute of Technology, Cambridge, MA.

Watson J-P, Wets RJ-B, Woodruff DL (2010) Scalable heuristics for a class of chance-constrained stochastic programs. *INFORMS J. Comput.* 22(4):543–554.

Zhang P (1996) Nonparametric importance sampling. *J. Amer. Statist. Assoc.* 91(435):1245–1253.