Imperial College London
Department of Computing

Importance Sampling for Stochastic Programming

Quang Kha Tran

supervised by

Dr. Panos Parpas
Prof. Berç Rustem

Submitted in partial fulfilment of the requirements for the degree of Doctor of Philosophy in Computing of Imperial College and the Diploma of Imperial College, 2016

DECLARATION OF ORIGINALITY

I, Quang Kha Tran, confirm that this thesis is my own work and that any additional sources of information have been referenced appropriately.

COPYRIGHT DECLARATION

Abstract

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 exptected 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 function evaluations to produce accurate results for large-scale problems with multiple periods and high-dimensional uncertainty. In this thesis, 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 modelled 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 low-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-to-high variance distributions or rare-event distributions. For example, in some applications, we found that if the random variables are drawn from a rare-event distribution, our proposed algorithm can achieve four times reduction in the mean square error and variance given by other existing methods (e.g.: SDDP with Crude Monte Carlo or SDDP with Quasi Monte Carlo method) for the same number of samples. Or when the random variables are drawn from the high variance distribution, our proposed algorithm can reduce the variance averagely by two times compared

to the results obtained by other methods for approximately the same level of mean square error and a fixed number of samples.

We use our proposed algorithm to solve a capacity expansion planning problem in the electric power industry. The model includes the unit commitment problem and maintenance scheduling. It allows the investors to make optimal decisions on the capacity and the type of generators to build in order to minimize the capital cost and operating cost over a long period of time. Our model computes the optimal schedule for each of the generators while meeting the demand and respecting the engineering constraints of each generator. We use an aggregation method to group generators of similar features, in order to reduce the problem size. The numerical experiment shows that by clustering the generators of the same technology with similar size together and apply the SDDP algorithm with our proposed sampling framework on this simplified formulation, we are able to solve the problem using only one fourth the amount of time to solve the original problem by conventional algorithms. The speed-up is achieved without a significant reduction in the quality of the solution.

Acknowledgement

I owe my deepest gratitude to Professor Berç Rustem and Dr. Panos Parpas at Imperial College London for their invaluable suggestions and advice for my Ph.D. thesis. Their commitments to excellence are inspiring and their enthusiasms for optimization are exceptional. Also, I would like to express my sincere gratitude to Professor Mort Webster in the Pennsylvania State University and my colleague Berk Ustun at Massachusetts Institute of Technology for many useful discussions and hours of programming.

I am deeply indebted to my family for all their unconditional love, support and encouragements.

Last but not least, I would like to thank all of the members in the Computational Optimisation Group at Imperial College London: Professor Daniel Kuhn, Dr. Wolfram Wiesemann, Dr. Ruth Misener, Dr. Duy Luong, Dr. Grani Adiwena Hanasusanto, Dr. Vladimir Roitch, Chin Pang Ho, Sei Howe, Vahan Hovhannisyan, Juan Campos Salazar, Georgia Kouyialis, Radu Baltean-Lugojan.

Contents

List of Figures

List of Tables

Chapter 1

Introduction to Linear Stochastic Programming and thesis overview

1.1 Overview of the thesis

Stochastic programming (SP) is a framework for modelling large-scale optimization problems that involve uncertainty. It has been applied effectively to solve many real-life problems in energy, finance and engineering. Solving a stochastic program, however, is usually very difficult because it requires the computation of the recourse function, which is usually a very high-dimenisonal integral, where each integrand is an optimization problem. The ultimate aim of this thesis is to develop an efficient and robust optimization algorithm for solving SP problems. We focus on large scale multistage linear stochastic programs because of their many applications but some of the proposed methodology can be extended to other classes of stochastic programs. Our proposed algorithm addresses the issue of efficient uncertainty sampling for stochastic programs. The benefits of the proposed algorithm will be demonstrated by solving a large number of benchmark test problems with various sizes, structures, time periods and nature of uncertainty. After that, we provide the theoretical convergence proof for the algorithm. Finally, the proposed algorithm is used to solve a very large-scale optimization problem in the eletric power industry, in which the unit commitment problem, maintenance scheduling and capacity expansion planning are integrated into a single model to allow the utilities investors make optimal decisions on the capacity expansion for different type of generators. The thesis consists of three main chapters:

1. In Chapter 3, we propose a new sampling algorithm, integrate it into an optimization algorithm and test it on several benchmark test problems

2. In Chapter 4, we provide a convergence proof for the proposed algorithm

3. In Chapter 6, we apply the new algorithm on a large-scale optimization problem in the power industry.

In the next section, we will introduce the SP framework. We will explain the computational challenges for solving this kind of problems and review some existing methods. After that, we propose an algorithm for solving SP problems with an increased efficiency and accuracy.

1.2 Linear Stochastic Programming

In this section, we will first introduce the SP framework. We will start with a two-stage stochastic linear program and then extend it to the multistage problem.

The two-stage stochastic linear program with fixed recourse was first introduced by Dantzig (1955) and Beale (1955) and it has the following form:

$$\begin{aligned} \min \quad & c^T x + E_\xi\{\min q(w)^T y(w)\} \\ \text{s.t.} \quad & Ax = b \\ & T(w)x + Wy(w) = h(w) \\ & x \geq 0, y(w) \geq 0 \end{aligned} \tag{1.1}$$

Where $x \in \mathbb{R}^{n_1}$ represents the first-stage decisions; $c \in \mathbb{R}^{n_1}$ is the vector associated with the first-stage objective function. $A \in \mathbb{R}^{m_1 \times n_1}$, $b \in \mathbb{R}^{m_1}$ are matrix and vector associated with the first-stage constraints. At the second stage, there are a number of random events happening. Each random event is denoted as $w \in \Omega$. For a given realization w, we obtain the matrices $q(w)$, $h(w)$ and $T(w)$, where $q(w) \in \mathbb{R}^{n_2 \times 1}$, $h(w) \in \mathbb{R}^{m_2 \times 1}$ and $T(w) \in \mathbb{R}^{m_2 \times n_1}$. The stochastic components of the problem are then grouped into $\xi^T(w)$, where $\xi^T(w) := (q(w)^T, h(w)^T, T_1(w), \ldots, T_{m_2}(w))$ and $T_i(w)$ is the ith row of the matrix $T(w)$. Thus $\xi^T(w)$ has $N = n_2 + m_2 + (m_2 \times n_1)$ components. For ease of exposition, we will use ξ and $\xi(w)$ interchangeably in this thesis. We assume that ξ has its support over $\Xi \subset \mathbb{R}^N$ such that $P(\Xi) = 1$. The vector $y(w) \in \mathbb{R}^{n_2}$ represents the second-stage recourse decision. The notation $y(w)$ shows that the value of $y(w)$ will change with respect to each realization of w. The matrix W is called the *recourse matrix* and assumed to be fixed

so that the feasibility set of the problem can be computed in a convenient manner. E_ξ is the mathematical expectation with respect to the probability distribution of ξ.

The Deterministic Equivalent Program (DEP) is defined as:

$$\begin{aligned} \min \quad & c^T x + \mathcal{Q}(x) \\ \text{s.t.} \quad & Ax = b \\ & x \geq 0 \end{aligned} \tag{1.2}$$

Where,

$$\mathcal{Q}(x) = E_\xi\{Q(x, \xi(w))\} \tag{1.3}$$

$$\begin{aligned} Q(x, \xi(w)) = \quad \min_y \quad & \{q(w)^T y(w)\} \\ \text{s.t.} \quad & Wy(w) = h(w) - T(w)x \\ & y(w) \geq 0 \end{aligned} \tag{1.4}$$

Equation (1.2) is known as the first-stage problem. Notice that the formulation (1.1)-(1.4) can be applied to both discrete and continuous random variables. $\mathcal{Q}(x)$ is known as the *recourse function*, which is the expected value of all second-stage (future cost) functions $Q(x, \xi(w))$.

In many practical situations, decision makers have to make decisions over a long period of time (more than two time-periods). In this case, equation (1.1) is extended to the multistage problem as follows:

$$\begin{aligned} \min \quad & {c_1}^T x_1 + E_{\xi_2}\{\min {c_2(w)}^T x_2(w_2) + \ldots + E_{\xi_H}\{\min {c_H(w)}^T x_H(w_H)\}\} \\ \text{s.t.} \quad & W_1 x_1 = h_1 \\ & T_1(w)x_1 + W_2 x_2(w_2) = h_2(w) \\ & \vdots \\ & T_{H-1}(w)x_{H-1}(w_{H-1}) + W_H x_H(w_H) = h_H(w) \\ & x_1 \geq 0, x_t(w_t) \geq 0, t = 2, \ldots, H \end{aligned} \tag{1.5}$$

Where $c_1 \in \mathbb{R}^{n_1}$, $W_1 \in \mathbb{R}^{m_1 \times n_1}$, $h_1 \in \mathbb{R}^{m_1}$ are deterministic vectors. $W_t \in \mathbb{R}^{m_t \times n_t}$ is a known matrix and $c_t(w)$, $h_t(w)$, $T_t(w)$ are grouped into $\xi_t(w)$ such that $\xi_t(w)^T := (c_t(w)^T, h_t(w)^T, T_1^{t-1}(w), \ldots, T_{m_t}^{t-1}(w))$. H is the total number of stages (time-periods). The decisions x depend on the history up to time t, which is indicated by w_t. We suppose that Ξ_t is the support of ξ_t.

The multistage problem (1.5) can be formulated as a Dynamic Program

(DP). In this case, the problem at the terminal stage is given as:

$$\begin{aligned} Q_H(x_{H-1}, \xi_H(w)) = \min \quad & c_H(w)^T x_H(w) \\ \text{s.t.} \quad & W_H x_H(w) = h_H(w) - T_{H-1}(w) x_{H-1} \\ & x_H(w) \geq 0 \end{aligned} \tag{1.6}$$

Let $\mathcal{Q}_{t+1}(x_t) = E_{\xi_{t+1}}\{Q_{t+1}(x_t, \xi_{t+1}(w))\}$ for all t. The recursive problem at stage $t = H-1, H-2, \ldots, 2$:

$$\begin{aligned} Q_t(x_{t-1}, \xi_t(w)) = \min \quad & c_t(w)^T x_t(w) + \mathcal{Q}_{t+1}(x_t) \\ \text{s.t.} \quad & W_t x_t(w) = h_t(w) - T_{t-1}(w) x_{t-1} \\ & x_t(w) \geq 0 \end{aligned} \tag{1.7}$$

Where x_t is the state of the system. The value that we aim to compute is:

$$\begin{aligned} \min \quad & {c_1}^T x_1 + \mathcal{Q}(x_1) \\ \text{s.t.} \quad & W_1 x_1 = h_1 \\ & x_1 \geq 0 \end{aligned} \tag{1.8}$$

Which has the same form of the two-stage stochastic linear programming problem (1.1). In the next section, we are going to explain some challenges of solving the multistage SP problems described above.

1.3 Computational Challenges

Multistage SP problems usually have thousands or millions of variables at every stage, corresponding to thousands or millions of random events that can occur at different points in time. This problem is known as the "curse of dimensionality". The curse of dimensionality means that the computational effort required to solve a stochastic program grows exponentially with the number of dimensions of the stochastic program. As a result, many people are naturally inclined to solve simpler versions. For example, all random variables are replaced with their expected value, so the problem is translated back to a deterministic form that can be easily solved. However, this approach may give us a wrong decision due to the fact that the nature of random variables cannot be fully represented by only a single value. Another approach for solving SP is by using scenario analysis introduced by Raiffa and Schalaifer (1961). In this approach, uncertainty is modelled through a number of scenarios and each scenario corresponds to a deterministic optimization problem. The optimal solution is the expected value of all scenarios'

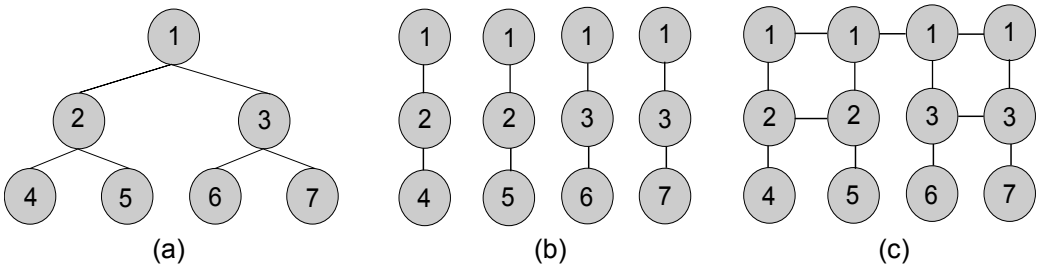


Figure 1.1: Different approaches to solve a stochastic program. (a) Compact formulation: Decomposition Method (b) Split-view formulation (c) Generating constraints to make the solutions of subproblems consistent after split-view

objective values. This is called the *wait-and-see* solution. Wait-and-see solution, however, cannot be implemented in practice because it assumes that decision makers can predict the future. Therefore, the ultimate aim is to use SP with some special algorithms that can exploit the structure of SP models, to make the calculation more efficient and tractable.

1.4 Existing algorithms

There are several approaches for solving SP problems (Parpas and Rustem (2007)). The first approach is to use the compact (recursive) formulation. With this approach, the formulation can be mapped directly onto a tree structure known as scenario tree (Figure 1.1(a)). The root of tree represents the current state of the world. Each subsequent level represents a future time period. At each time period, there are different nodes representing the different events that may occur. The path from the root to a leaf node is called a scenario. The multistage SP problem is then solved by a nested linear program using decomposition algorithms such as *Bender's decomposition* (L-shaped method). In Benders decomposition, the large-scale problem is decomposed into several subproblems, each representing a node in the tree.

The second approach for solving SP problems is to use the split-view formulation (Figure 1.1(b)). With this approach, a large-scale problem is split into n subproblems, where n is the number of scenarios. Because each subproblem is solved separately, new constraints that are commonly known as the non-anticipative constraints are introduced to make the solutions consistent across different subproblems (Figure 1.1(c)). Some examples of the split-view formulation are the Augmented Lagrangian method, first intro-

duced by Hestenes (1969) and Powell (1969) and the Progressive Hedging algorithm, first introduced by Rockafellar and Wets (1991).

The drawbacks of these approaches are that they require the construction of a scenario tree. Constructing a scenario tree can become a very complicated task as the complexity of the scenario tree grows exponentially with the number of stages and random variables. Therefore, only a small scenario tree tends to be used. With a small number of scenarios, it is possible that we cannot obtain the true optimal solution. This problem occurs not only in the situation when continuous random variables are discretized via sampling, but also in the situation when there are so many discrete random variables (possible scenarios) that we need to perform some sampling.

Another approach for solving multistage SP problems is to use the Linear Decision Rules method (Kuhn et al. (2011)). A great advantage of the Linear Decision Rules method is it can reduce significantly the computational complexity. The decisions that are desired to obtain, however, have to follow the linear relationship with the data. This leads to a considerable loss of accuracy and limits the applicability of the Linear Decision Rules method in many situations.

SP problems can also be solved by sampling algorithms such as the Sample Average Approximation (SAA) method Kleywegt et al. (2001). It is shown that SAA can solve a two-stage linear SP very efficiently using the Monte Carlo sampling technique. The obtained results are the approximation to the true solutions with a reasonable accuracy. It is, however, shown that SAA cannot solve a multi-stage SP due to the "curse of dimensionality" (Shapiro and Nemirovski (2005)).

As a result, the primary aim of this thesis is to devise an efficient and accurate algorithm for multistage SP problems. In the next section, we will explain the plan for achieving it.

1.5 Proposed Methodology

Our primary aim in this thesis is to devise an efficient and accurate algorithm for multistage SP problems. In order to achieve this aim, we start with the analysis of one of the state-of-the-art algorithms for solving multistage SP problems. The algorithm is called the Stochastic Dual Dynamic Programming (SDDP) method, first introduced by Pereira and Pinto (1991). SDDP is essentially the combination of SAA and Benders decomposition. SDDP has many attractive features. Firstly, it does not require the construction of scenario trees; secondly, by exploiting the piecewise linear approximation of the recourse function, it constructs valid lower and upper bounds, which can

be used as the stopping criterion. One common drawback of sampling algorithms is that we need to take a lot of samples in order to achieve reasonably accurate results. Generating more samples, in practice, is a very expensive approach as each sample requires the solution of a Linear Program (LP). Therefore, our aim is to achieve a good approximation without increasing the number of samples. This is exactly the purpose of using Variance Reduction (VR) methods. Some commonly used VR methods are Control Variates, Antithetic Variates, Stratification and Importance Sampling.

Control Variates compute the approximate error of a known function (Control Variate) and use it to reduce the error of the unknown function. It is, however, not easy to find a suitable Control Variate that has a large correlated coefficient (as close to one as possible) with the unknown function. In practice, one can try to make the Control Variate depend on the unknown function as much as possible but this is still a very difficult approach to generalize.

Antithetic Variates reduce the estimation error by generating negatively correlated samples with the obtained samples. The new generated samples, hence, form with the obtained samples to become a pair with negative correlation. Because of this negative coefficient, the overall variance can be reduced. However, there are not many realistic examples in which the variance reduction obtained by Antithetic Sampling is dramatic. Also, with this approach, we still have to take additional samples, which is not our aim.

Stratification reduces the estimation error by dividing the set of samples into several homogeneous subset of samples, called Strata. A common strategy of Stratification is proportional allocation, in which the size of each Strata should be proportional to the standard deviation of the distribution of the variables. Larger samples are taken in the stratum with the greatest variability to generate the least possible sampling variance. The disadvantage of Stratification is that the samples used in SP problems cannot be partitioned into disjoint subgroups and it is not possible to know the variance (or standard deviation) at each region in the sample space.

Importance Sampling (IS) reduces the estimation error by modifying the sampling distribution such that most of the samples are generated in the region that contribute the most to the function that is desired to calculate. Because the samples are generated in such a biased way, if we use this biased distribution directly in the simulation, the estimator will become biased. In order to correct this bias, the simulation output is weighted according to the likelihood ratio. This likelihood ratio is basically the ratio between the true underlying distribution with respect to the biased distribution. Therefore, by multiplying this ratio with the biased output, we are able to translate back to the original distribution and obtain an unbiased estimator. The fundamental

issue of IS is the choice of the biased distribution that biases the sampling towards the important regions. Choosing a good distribution can save a great amount of computational effort and make the approximation much more accurate by reducing the variance. In the context of SP problems, it reduces the number of LP problems needed to solve while the decreased variance in the approximate future cost function allows the optimization algorithm to improve its convergence rate and the solutions' accuracy. We will demonstrate an example of how a conventional Monte-Carlo sampling method with a high variance and small number of samples can give wrong solutions for an optimization problem in Section 2.7. Furthermore, IS is relatively easy to implement once we know the importance sampling distribution.

Importance Sampling has been applied to SP problems by Dantzig and Glynn (1990) and Infanger (1992). The method has shown a great advantage in reducing the variance of the recourse function approximation (Infanger (1992) Higle (1998)). Furthermore, it can capture rare events (e.g.: a power outage or a surge in demand in the electric power industry) much more effective than those using the crude Monte Carlo (see Infanger (1992) and Chapter 6 of this thesis). However, there are some strict assumptions associated with the methods proposed in the existing literature. For example, the random variables in the model can only be drawn from a discrete distribution. To our knowledge, the method has not been extended to the continuous case yet. Another important assumption is that the recourse function has to be additively separable with respect to the random variables. It has been shown that these assumptions have made the algorithm difficult to use in practice, and in some situations the variance of the obtained estimator is increased more than the crude Monte Carlo methods (Homem-de Mello and Bayraksan (2014) Higle (1998) Parpas et al. (2014)). In this thesis, we are going to propose an IS method that can be used to solve multistage SP problems without making any of the assumptions above.

The proposed IS exploits a well-known theory in the IS literature that there exists an optimal IS distribution that can give a zero-variance estimator (Asmussen and Glynn (2007)). Althouth this optimal IS distribution cannot be used in practice due to the *Curse of Circularity* [1], it shows that the optimal IS distribution is known up to a normalizing constant. We therefore propose to use Markov chain Monte Carlo (MCMC) to generate a sequence of samples (an ergodic Markov chain) such that after a sufficient long run, the samples generated by MCMC will converge to a stationary distribution that is very close to the optimal IS distribution for computing the recourse function of the

[1] It is possible to find an optimal sample (and therefore zero variance) but the optimal solution needs to be known!

SP problems. After that, we propose to use the Kernel Density Estimation (KDE) to reconstruct the approximately optimal IS distribution. This step is important for two reasons. Firstly, more samples can be generated from the obtained optimal IS distribution with little marginal computational cost while offering a better approximation. Secondly, the obtained optimal IS density is used to calculate the likelihood ratio, which then corrects the bias in the IS estimator.

The advantages of the proposed IS framework will be demonstrated throughout the thesis. Here are its main features. Firstly, it can work well with many different types of MCMC and KDE algorithms. The continuous advancement of MCMC or KDE will, to some extent, lead to the improvement of the proposed IS. Secondly, the proposed IS method shares the same benefit with many other existing IS methods such that it can give robust and reasonable results for the probability distributions that are difficult to work with. Finally, it is well-suited for SP optimization algorithms because it obtains lower-variance estimates of the recourse function that eventually helps to solve the models more accurately.

The proposed IS framework is a sampling method and it has to be paired with an optimization algorithm to solve SP problems. The optimization algorithm that we use in this thesis is called the Stochastic Dual Dynamic Programming (SDDP) method (Pereira and Pinto (1991)) because of many attractive features mentioned at the beginning of this section. The SDDP algorithm will be explained in Section 2.6. On the other hand, we expect that the proposed IS framework is flexible and simple enough for it to work with many other stochastic optimization algorithms such as the SAA (Shapiro et al. (2009)), Progressive Hedging (Rockafellar and Wets (1991)), Stochastic Decomposition (Higle and Sen (1991)), Augmented Lagrangian methods (Parpas and Rustem (2007)), Approximate Dynamic Programming (Powell (2007)) and some variants of Benders' decomposition methods (Birge and Louveaux (2011)).

1.6 Thesis outline

In Chapter 2, we will explain the principles of different methods for solving multistage stochastic programming problems, including the Bender's decomposition method and the Stochastic Dual Dynamic Programming method. We will show the error associated with sampling methods, and then review different approaches that have been used to solve this problem.

In Chapter 3, we will propose an efficient and accurate algorithm to solve multistage stochastic programming problems. The proposed sampling al-

gorithm is based on the theory of Importance Sampling while using the techniques of Markov chain Monte Carlo and Kernel Density Estimation. Therefore, we will review each technique first before explaining the way to apply them for stochastic programming problems. The proposed sampling algorithm is then paired with a stochastic optimization algorithm such as Stochastic Dual Dynamic Programming, Sample Average Approximation, and so on, in order to solve multistage stochastic programming problems.

In Chapter 4, we will prove the convergence of the proposed algorithm. To do so, we need to show the convergence of the Markov Chain Monte Carlo that we use in the proposed algorithm. After that, we show the convergence of the Kernel Density Estimation for the generated Markov chain. Finally, we show the convergence of the stochastic optimization algorithm (e.g.: Stochastic Dual Dynamic Programming) when it uses the proposed sampling algorithm.

In Chapter 5, we will firstly test the proposed algorithm on the Newsvendor problem in order to understand the properties of the algorithm. Then, we use the proposed algorithm to solve a wide range of applications, which are the benchmark stochastic programming test problems found in Ariyawansa and Felt (2004). We compare the results obtained by the proposed algorithm with some existing methods such as the Crude Monte Carlo, Quasi Monte Carlo and the Importance Sampling method proposed by Infanger (1992).

In Chapter 6, we introduce the capacity expansion planning model in the electric power industry. The model also includes the unit commitment problem and maintenance scheduling and it is well-known to be a very large scale optimization problem. We propose different formulations to reduce the problem size and make the problem scale more efficiently with the number of time periods. Then we use the proposed algorithm to solve these formulations. The obtained results are compared with the ones obtained by the Stochastic Dual Dynamic Programming method.

In Chapter 7, we present our conclusions and describe some possible future work.

1.7 Contributions

The contents of Chapter 3 were presented in the International Conference on Stochastic Programming, Bergamo, Italy in 2013. The resulting paper entitled: Importance Sampling in Stochastic Programming: A Markov Chain Monte Carlo Approach, has been accepted for publication in INFORMS Journal of Computing. The main contribution of this work is the development of an efficient and accurate algorithm for solving multistage stochastic pro-

gramming problems.

The contents of Chapter 6 are in preparation for the European Journal of Operational Research. The main contribution of this work is the development of the stochastic capacity expansion planning model for the electric power industry and using the proposed algorithm in Chapter 3 to solve it efficiently and accurately.

Chapter 2

Background

In this chapter, we are going to study different methods for solving multistage stochastic programming problems. The methods include Bender's decomposition (L-shaped method), Regularized decomposition, Basic Factorization, Interior Point methods, Dantzig-Wolfe decomposition (Inner Linearization approach), Basic Lagrangian Dual Ascent, Progressive Hedging and the Stochastic Dual Dynamic Programming algorithm. We then show the approximation error associated with these sampling algorithms. After that, we investigate different ways to solve this problem.

2.1 Bender's decomposition

Bender's decomposition (L-shaped method) is an algorithm that exploits the special structure in SP problems in order to solve them efficiently Benders (1962). The basic idea of Bender's decomposition is that it splits the original two-stage stochastic linear program into a master problem and many independent subproblems. Each subproblem corresponds to a scenario $w \in \Omega$. By solving the master problem, the first-stage decisions x is obtained. Given x, the subproblems are solved to generate a supporting hyperplane, which is well-known as a cut. This cut is used to approximate the recourse function and it is added to the master problem. Then solving the master problem gives us another trial decision. The algorithm repeats until the optimality criterion is satisfied. In this section, we will derive the main steps of Bender's decomposition algorithm for a two-stage stochastic LP problem.

Assume that the random vector ξ has finite support. Let $k = 1, \ldots, K$ be the index of all possible realizations and p_k be their probabilities. Under this assumption, the second-stage decisions y_k is associated with a realization $\xi_k(w)$, i.e. to each realization of $q_k(w)$, $h_k(w)$ and $T_k(w)$. As a result,

equation (1.1) can be rewritten in the extensive form as follows:

$$\begin{aligned} \min_{x,y_k(w)} \quad & c^T x + \sum_{k=1}^{K} p_k q_k(w)^T y_k(w) \\ \text{s.t.} \quad & Ax = b \\ & T_k(w)x + W y_k(w) = h_k(w), k = 1, \ldots, K \\ & x \geq 0, y_k(w) \geq 0, k = 1, \ldots, K \end{aligned} \tag{2.1}$$

The projected problem of Equation (2.1) is given as:

$$\min_{x \in V} c^T x + \sum_{k=1}^{K} p_k v_k(x) \tag{2.2}$$

Where

$$V = \{x \in X | Ax = b, \exists y_k(w) \in Y, T_k(w)x + W y_k(w) = h_k(w)\}$$

and

$$\begin{aligned} v_k(x) = \min_{y_k(w)} \quad & q_k(w)^T y_k(w) \\ \text{s.t.} \quad & W y_k(w) = h_k(w) - T_k(w)x \end{aligned} \tag{2.3}$$

Therefore, an equivalent formulation to Equation (2.2) is given as:

$$\begin{aligned} \min_{x \in V} \quad & c^T x + \theta \\ \text{s.t.} \quad & \theta \geq \sum_{k=1}^{K} p_k v_k(x) \end{aligned} \tag{2.4}$$

In addition, the dual of Equation (2.3) is given by:

$$\begin{aligned} v_k(x) = \max_{\pi_k(w)} \quad & \pi_k(w)^T \{h_k(w) - T_k(w)x\} \\ \text{s.t.} \quad & W^T \pi_k(w) \leq q_k(w) \end{aligned} \tag{2.5}$$

Assumed that $v_k(x)$ is differentiable, and since $v_k(x)$ is convex in x, the subgradient inequality can be described as:

$$v_k(x) \geq v_k(x_s) + d_k^T (x - x_s) \tag{2.6}$$

Where $d_k \in \partial v_k(x_s)$ and $v_k(x_s)$ is the subgradient of v_k at x_s. Substituting equation (2.6) into equation (2.4),

$$\begin{aligned} \min_{x \in V} \quad & c^T x + \theta \\ \text{s.t.} \quad & \theta \geq \sum_{k=1}^{K} p_k \{v_k(x_s) + \nabla v_k(x_s)^T (x - x_s)\} \end{aligned} \tag{2.7}$$

From equation (2.5), the subgradient $\nabla v_k(x_s)$ is given as:

$$\nabla v_k(x_s) = -\pi_k(w)^T T_k(w) \tag{2.8}$$

Where

$$\pi_k(w) \in \arg \max_{W^T \pi_k(w) \leq q_k(w)} \pi_k(w)^T (h_k(w) - T_k(w)x)$$

Substituting (2.5), (2.8) back into equation (2.7), one obtains:

$$\begin{aligned} \min_{x \in V} \quad & c^T x + \theta \\ \text{s.t.} \quad \theta \quad & \geq \sum_{k=1}^{K} p_k \{\pi_k(w)^T (h_k(w) - T_k(w)x_s) - \pi_k(w)^T T_k(w)(x - x_s)\} \\ & \geq \sum_{k=1}^{K} p_k \{\pi_k(w)^T h_k(w) - \pi_k(w)^T T_k(w)x\} \end{aligned} \tag{2.9}$$

This is known as the *optimality cut* of Bender's decomposition.

So far, it has been assumed that the subproblem (2.3) always has feasible solutions for every x and w. This is not always the case. In general, one needs to check the feasibility of subproblem by solving the following problem:

$$\begin{aligned} I_k(x) = \min \quad & e^T z^+ + e^T z^- \\ \text{s.t.} \quad & W y_k(w) + I z^+ + I z^- = h_k(w) - T_k(w)x \\ & y_k(w) \geq 0, z^+ \geq 0, z^- \geq 0 \end{aligned} \tag{2.10}$$

Where $e^T = [1, \ldots, 1]$ and I is the identity matrix.

If x is feasible, $I_k(x) = 0$; otherwise, $I_k(x) > 0$. The dual of equation (2.10) is given as:

$$\begin{aligned} I_k(x) = \max_{\pi_k(w)} \quad & \pi_k(w)^T (h_k(w) - T_k(w)x) \\ \text{s.t.} \quad & W^T \pi_k(w) \leq 0 \\ & -1 \leq \pi_k(w) \leq 1 \end{aligned} \tag{2.11}$$

Let π^* be the vector solution of (2.5). Then, by adding the following constraint

$$\pi^{*T} (h_k(w) - T_k(w)x) \leq 0 \tag{2.12}$$

to the master problem, it removes all infeasible solutions while keeping feasible solutions. This is because: for any infeasible x_{infeas}, $\pi^{*T}(h_k(w) - T_k(w)x_{\text{infeas}}) = I_k(x_{\text{infeas}}) > 0$; and for any feasible x_{feas}, $\pi^{*T}(h_k(w) - T_k(w)x_{\text{feas}}) \leq I_k(x_{\text{inf}}) = 0$. Equation (2.12) is known as the *feasibility cut*.

The algorithm of Bender's decomposition is summarized in Algorithm 1. Bender's decomposition can be easily extended to multistage problems. At each stage, the algorithm solves subproblems for every realization of w, and add optimality or feasibility cuts to the previous-stage problem. The algorithm repeats until we can find the optimal first-stage decisions. An advantage of Bender's decomposition is that it can be used in parallel computing. The drawback of Bender's decomposition is that it requires the construction of scenario tree. The complexity of scenario tree grows exponentially with the number of random variables (samples) and number of stages (time periods). This is known as the *Curse of Dimensionality*. To avoid this problem, a small scenario tree tends to be used. However, with only a small number of samples, we sometimes cannot obtain the true optimal solution. This problem not only occurs in the situation when continuous random variables are discretized by sampling algorithms, but also in discrete cases, when there are so many discrete values that we need to perform some sampling. In any cases, constructing a scenario tree is cumbersome. In Section 2.6, we will explain a methodology called Stochastic Dual Dynamic Programming (SDDP), which allows us to bypass the scenario tree requirement and provide an efficient approach of solving multistage SP problems while maintaining the parallel computing possibilities.

2.2 Regularized decomposition

Regularized decomposition was first introduced by Ruszczynski (1986). It includes a quadratic regularizing term in the objective function, which allows the algorithm to leverage good starting points. Moreover, it generates a cut for each realization, as opposed to solving all of the subproblems for all realizations in order to generate only a cut as in Bender's decomposition. The benefit of adding more cuts at each iteration is the algorithm may take fewer iterations to converge. On the other hand, the problem now has a larger number of constraints so the computational cost for each iteration increases, which may cancel out the advantage of having fewer iterations. As a result, the successful use of regularized decomposition depends on the problem (Birge and Louveaux (1988)) and (Gassman (1990)). The rule-of-thumb is that the multicut approach is expected to be more effective when the number of realizations are not much larger than the number of first-stage constraints.

Algorithm 1 Bender's decomposition Algorithm

1. Initialize. Set $n_{feas} = 0, n_{opt} = 0, i = 0$ (where n_{feas} is the number of feasibility cuts, n_{opt} is the number of optimality cuts, i is the number of iterations).
2. Solve the master problem

$$\begin{aligned} \min \quad & c^T x + \theta \\ \text{s.t.} \quad & Ax = b \\ & d_l - D_l x \leq 0, l = 1, \ldots, n_{feas} \text{ (feasibility cuts as in Equation (2.12))} \\ & \theta \geq e_l - E_l x, l = 1, \ldots, n_{opt} \text{ (optimality cuts as in Equation (2.9))} \\ & x \geq 0 \end{aligned} \tag{2.13}$$

Let (x^i, θ^i) be the optimal solution at i-th iteration. If there is no optimality cuts, θ^i is simply not considered and set to $-\infty$.
If the solutions converge, STOP. Otherwise, go to step 3.
3. For $k = 1, \ldots, K$, solve the following subproblem:

$$\begin{aligned} \min \quad & q_k(w)^T y(w) \\ \text{s.t.} \quad & W_k(w) y(w) = h_k(w) - T_k(w) x^i \\ & y \geq 0 \end{aligned} \tag{2.14}$$

If all the problems are feasible, add the following optimality cut to the master problem:

$$\theta \geq e_l - E_l x \tag{2.15}$$

Where $e_l = \sum_{k=1}^{K} p_k (\pi_k^i)^T (h_k(w))$; $E_l = \sum_{k=1}^{K} p_k (\pi_k^i)^T (T_k(w))$; π_k^i is the multiplier of (2.14). Set $n_{opt} = n_{opt} + 1$.
If Equation (2.14) is infeasible, solve the problem (2.10); and then add the following feasibility cut to the master problem:

$$d_l - D_l x \leq 0$$

Where $d_l = (\sigma^i)^T (h_k(w))$; $D_l = (\sigma^i)^T (T_k(w))$ and σ^i is the multiplier of Equation (2.10).
Return to step 2

2.3 Dantzig-Wolfe decomposition

The Dantzig-Wolfe decomposition method (Inner Linearization method) was first introduced by Dantzig and Wolfe (1960). It is the dual of the Bender's decomposition method that we have mentioned in Section 2.1. Therefore, instead of solving the problem (2.13), we solve the following problem:

$$\begin{aligned} \max \quad & \rho^T b + \sum_{l=1}^{r} \sigma_l d_l + \sum_{l=1}^{s} \pi_l e_l \\ \text{s.t.} \quad & \rho^T A + \sum_{l=1}^{r} \sigma_l D_l + \sum_{l=1}^{s} \pi_l E_l \leq c^T \\ & \sum_{l=1}^{s} \pi_l = 1 \\ & \sigma_l \geq 0, l = 1, \ldots, r \\ & \pi_l \geq 0, l = 1, \ldots, s \end{aligned} \tag{2.16}$$

It can be seen that problem (2.16) is the dual problem of the linear program (2.13). However, it is the master problem in the Dantzig-Wolfe decomposition method. Once we solve it, we will obtain the solutions (ρ, σ, π) and the duals (x, θ), which are then used to solve the subproblem given as follows:

$$\begin{aligned} \max \quad & \pi_k^T (h_k(w) - T_k(w)x) \\ \text{s.t.} \quad & W_k(w)^T \pi_k \leq q_k(w) \end{aligned} \tag{2.17}$$

If the problem (2.17) is infeasible for any realization, it generates a *feasibility* cut and adds this cut in the *column* corresponding to $d_{r+1} = \sigma^T h_k$ and $D_{r+1} = \sigma^T T_k$. If all problems are feasible, it generates a *optimality* cut and adds this cut in the *column* corresponding to $e_{s+1} = \sum_{k=1}^{K} p_k \pi_k^T h_k$ and $E_{s+1} = \sum_{k=1}^{K} p_k \pi_k^T T_k$. In the same way as the Bender's decomposition method, the algorithm terminates when $\theta \geq e_{s+1} - E_{s+1}x$.

In practice, many problems have the first-stage decision variables with a large number of dimensions and a few number of constraints. This means that the primal version has a smaller basis matrix than the basis of the dual (Birge and Louveaux (2011)). As a result, the Bender's decomposition method is more effective than the Dantzig-Wolfe decomposition in many situations.

2.4 Basic Lagrangian Dual Ascent and Augmented Lagrangian

Consider the following problem:

$$\begin{aligned} \inf \quad & f_1(x) + \mathcal{Q}(x) \\ \text{s.t.} \quad & g_{1,i}(x) \leq 0, i = 1, \ldots, m_1 \end{aligned} \tag{2.18}$$

where $\mathcal{Q}(x) = E_\xi\{Q(x, y(w), \xi(w))\}$ and

$$\begin{aligned} Q(x, y(w), \xi(w)) = \quad & \inf \quad f_2(x, y(w), \xi(w)) \\ & \text{s.t.} \quad g_{2,i}(x, y(w), \xi(w)) \leq 0, i = 1, \ldots, m_2 \end{aligned}$$

where $f_1(x)$ and $f_2(x)$ are the objective functions of the first and second stage; $g_{1,i}(x)$ and $g_{2,i}(x)$ are the constraints for the first and second-stage problem; m_1 is the number of constraints in the first stage, m_2 is the number of constraints in the second stage. The idea of Lagrangian methods is to put any links between the first and second-stage variables into the objective, instead of having the hard constraints in the problem. In other words, the Lagrangian methods solve problem (2.18) by reformulating it as follows:

$$\max_{\pi(w) \geq 0} \theta(\pi(w)) \tag{2.19}$$

where

$$\begin{aligned} \theta(\pi(w)) = \inf_{x, y(w)} \quad & f_1(x) + E_\xi\{f_2(x, y(w), \xi(w))\} \\ & + E_{\xi(w)}\{\sum_{i=1}^{m_2} \pi_i(w) g_{2,i}(x, y(w), \xi(w))\} \\ \text{s.t.} \quad & g_{1,i}(x) \leq 0, i = 1, \ldots, m_1 \end{aligned} \tag{2.20}$$

Suppose that the problem 2.18 has a finite optimal value, all of the functions are convex and there is a point strictly satisfying all constraints, the Basic Lagrangian Dual Ascent method will iterate between two main steps. The first step is to solve the problem (2.20) in order to obtain the approximate solutions for $\{x, y(w)\}$. The second step is to solve the problem (2.19) i.e. to find the next value of $\pi(w)$ that maximizes the function θ. The algorithm will converge to optimal solutions that are equivalent to the solutions of problem (2.18). The efficiency of the algorithm depends on the computational cost of finding the dual in problem (2.19) relatively to the cost of solving the original problem (2.18).

The Augmented Lagrangian adds a penalty term $r\|g_{2,i}(x, \xi(w))^+\|^2$ to the objective function. This allows us to perform the Newton-type steps since the Hessian matrix now becomes nonsingular. As a result, the convergence of the algorithm can improve significantly (Dempster (1988)).

2.5 Progressive Hedging

The method was first introduced by Rockafellar and Wets (1991). In this algorithm, problem (2.1) is separated into several subproblems and each subproblem corresponds to a realization. The computational cost for a subproblem is therefore relatively small but the number of iterations for the algorithm convergence may increase. Similar to the Bender's decomposition method, the advantage of this method is the subproblems can be solved in parallel. The principles of the Progressive Hedging method are as follows: First, problem (2.18) is reformulated as:

$$\begin{aligned} \inf \quad & \sum_{k=1}^{K} p_k \{ f_1(x) + f_2(x, \xi(w)) + \rho(x - x_{prev}) + \frac{r}{2} \|x - x_{prev}\|^2 \} \\ \text{s.t.} \quad & g_{1,i}(x) \leq 0, \ i = 1, \ldots, m_1, \ k = 1, \ldots, K \\ & g_{2,i}(x, \xi(w)) \leq 0, \ i = 1, \ldots, m_2, \ k = 1, \ldots, K \end{aligned} \tag{2.21}$$

Where $\xi(w)$ have K realizations. The algorithm iterates between two main steps. The first step is to find the approximate solutions for x. The second step is to compute the new optimal value of ρ such that $\rho_{new} = \rho + (x - \bar{x})$, where $\bar{x}$ is the expected value of x calculated in the previous iteration. The algorithm stops when $x = x_{prev}$ and $\rho = \rho_{prev}$; at that time, the algorithm has converged and (x, ρ) are proved to be the optimal solutions (Rockafellar and Wets (1991)).

2.6 Stochastic Dual Dynamic Programming

In this section, we introduce a method called Stochastic Dual Dynamic Programming (SDDP) for solving multistage SP problems (Pereira and Pinto (1991)). A great advantage of SDDP is that it does not require the scenario tree construction. The algorithm is based on the theory of Bender's decomposition (as explained in Section 2.1), in which the recourse function is approximated by a piecewise linear function. These functions are obtained from the dual solutions of subproblems at each stage. We are going to start with the deterministic problem. Then it is followed by the extension to the stochastic case.

2.6.1 Deterministic Dual Dynamic Programming

The concepts of Dual Dynamic Programming (DDP) is illustrated with the following LP problem:

$$\begin{aligned} \min \quad & {c_1}^T x_1 + {c_2}^T x_2 \\ \text{s.t.} \quad & A_1 x_1 \geq b_1 \\ & E_1 x_1 + A_2 x_2 \geq b_2 \end{aligned} \tag{2.22}$$

where $c_1 \in \mathbb{R}^{n_1}$, $x_1 \in \mathbb{R}^{n_1}$, $c_2 \in \mathbb{R}^{n_2}$, $x_2 \in \mathbb{R}^{n_2}$, $A_1 \in \mathbb{R}^{m_1 \times n_1}$, $b_1 \in \mathbb{R}^{m_1}$, $E_1 \in \mathbb{R}^{m_2 \times n_1}$, $A_2 \in \mathbb{R}^{m_2 \times n_2}$, $b_2 \in \mathbb{R}^{m_2}$. Problem (2.22) can be interpreted as a two-stage problem. The first-stage problem is defined as:

$$\begin{aligned} \min \quad & {c_1}^T x_1 + \alpha_1(x_1) \\ \text{s.t.} \quad & A_1 x_1 \geq b_1 \end{aligned} \tag{2.23}$$

The second-stage problem is defined as:

$$\begin{aligned} \alpha_1(x_1) = \min \quad & {c_2}^T x_2 \\ \text{s.t.} \quad & A_2 x_2 \geq b_2 - E_1 x_1 \end{aligned} \tag{2.24}$$

In Dynamic Programming (DP), ${c_1}^T x_1$ represents the *current cost*; $\alpha_1(x_1)$ represents the *future cost*, which is a function of the first-stage decision variables x_1. DP algorithms construct the future cost function by discretizing x_1 into a set of trial values $\{\hat{x}_{1i}, i = 1, \ldots, n\}$ and then solving problem (2.24) for each trial value. The intermediate values of $\alpha_1(x_1)$ can be obtained by interpolating the neighbouring discretized states. Once the future cost function is constructed, problem (2.23) is solved again to find the optimal solutions and objective value.

DP has many attractive properties. Firstly, it can easily extend to multistage and stochastic problems. Secondly, it can solve nonlinear problems with relative ease. Its drawback is that the first-stage decisions need to be discretized into a very large number of values such that the future cost function can be constructed accurately. The computation therefore can become very intensive or even intractable. For example, if the vector x_1 has ten components and each component is discretized into five values, there are $5^{10} \approx 9.76$ million different combinations of x_1. This problem is well-known as the *Curse of Dimensionality*.

An approach to avoiding the *Curse of Dimensionality* is to approximate the future cost function by an analytical function rather than discretizing the first-stage decision variables. In this case, the future cost function is approximated by a piecewise linear function by taking the dual of Equation

(2.24), which is a similar technique used in Bender's decomposition (Section 2.1). The dual of Equation (2.24) is given as:

$$\begin{aligned} \alpha_1(x_1) = \max \quad & \pi^T(b_2 - E_1 x_1) \\ \text{s.t.} \quad & {A_2}^T \pi \leq c_2 \end{aligned} \tag{2.25}$$

Where π is the simplex multiplier vector associated with the constraint in (2.23). According to the LP theory, the optimal solution obtained from the original problem coincides with the one obtained from its dual. Hence, problem (2.24) and (2.25) are equivalent and both of them represent the future cost function. However, the constraint ${A_2}^T \pi \leq c_2$ in problem (2.25) is not dependent on x_1, so a set of possible solutions π can be obtained even before knowing the decision x_1.

Let $\Pi = \{\pi_1, \ldots, \pi_m\}$ be the set of all vertices of the constraint in (2.25). The future cost function can be found using the complete *enumeration* method:

$$\alpha_1(x_1) = \max\{\pi^i\}^T(b_2 - E_1 x_1), i = 1, \ldots, m \tag{2.26}$$

As a result, the equivalent problem to problem (2.24) is:

$$\begin{aligned} \alpha_1(x_1) = \min \quad & \alpha \\ \text{s.t.} \quad & \alpha \geq \{\pi^i\}^T(b_2 - E_1 x_1), i = 1, \ldots, m \end{aligned} \tag{2.27}$$

Where α is a scalar variable. Equation (2.27) shows that the future cost function can be described by a piecewise linear function of different components. Each component is a hyperplane defined by $\{\pi^i\}^T(b_2 - E_1 x_1)$. It is sufficient to use only the coefficients $\{\pi_i\}$ to construct the future cost function, and it is not required to discretize x_1. Finding all vertices $\{\pi_i\}$, however, may be very challenging. For many problems, we can use only a subset of these vertices. If $\hat{x}_{1i}$ is the trial first-stage decisions, the vertices can be obtained by solving the dual of the following problems:

$$\begin{aligned} \alpha_1(\hat{x}_{1i}) = \min \quad & {c_2}^T x_2 \\ \text{s.t.} \quad & A_2 x_2 \geq b_2 - E_1 \hat{x}_{1i} \end{aligned} \tag{2.28}$$

Let π_i be the vector multiplier of problem (2.28). π_i therefore belongs to the set Π. Given a set of n trial values $\{\hat{x}_{1i}, i = 1, \ldots, n\}$, one can obtain n associated multipliers $\pi_i, i = 1, \ldots, n$ by solving (2.28) for each trial value. The future cost function therefore can be approximated as:

$$\begin{aligned} \hat{\alpha}_1(x_1) = \min \quad & \alpha \\ \text{s.t.} \quad & \alpha \geq \{\pi_i\}^T(b_2 E_1 x_1), i = 1, \ldots, n \end{aligned} \tag{2.29}$$

Since only a subset of $\{\Pi\}$ are used to approximate the future cost function, Equation (2.29) is the lower bound of the true future cost function. After that, we can solve the first-stage problem:

$$\begin{aligned} z = \min \quad & {c_1}^T x_1 + \hat{\alpha}_1(x_1) \\ \text{s.t.} \quad & A_1 x_1 \geq b_1 \end{aligned} \tag{2.30}$$

Substituting (2.29) into (2.30),

$$\begin{aligned} z = \min \quad & {c_1}^T x_1 + \alpha \\ \text{s.t.} \quad & A_1 x_1 \geq b_1 \\ & \alpha \geq \{\pi_i\}^T (b_2 - E_1 x_1), i = 1, \ldots, n \end{aligned} \tag{2.31}$$

Equation (2.31) is the lower bound of the true optimal cost because the approximate future cost is the lower bound to the true future cost. The lower bound $\underline{z}$ is therefore calculated as:

$$\underline{z} = {c_1}^T \hat{x}_1 + \hat{\alpha} \tag{2.32}$$

Where $\hat{x_1}$ and $\hat{\alpha}$ are the solutions of Equation (2.31). The upper bound $\bar{z}$ is obtained by solving second-stage problem (2.28) for the trial first-stage decisions $\hat{x}_1$:

$$\bar{z} = {c_1}^T \hat{x}_1 + \alpha_1(\hat{x}_1) \tag{2.33}$$

$\bar{z}$ and $\underline{z}$ can be considered as the *actual* and *predicted cost* respectively. If $(\bar{z} - \underline{z}) \leq \epsilon$ for a small tolerance $\epsilon \geq 0$, the optimal actual and predicted cost are very close to each other. The problem is solved. Otherwise, a new set of trial decisions is used and the whole process repeats. The new set of trial decisions can be obtained from the previous iteration. The approximate future cost function in this case can be built upon the previous good candidates for the optimal solution.

In summary, DDP has several advantages. Firstly, it does not require state discretization (Pereira and Pinto (1991)). Secondly, the optimal solution obtained at every iteration can be reused as the trial solution for the next iteration. Thirdly, the upper and lower bound can be calculated for every iteration and they are used directly for the stopping criterion. Finally, the algorithm will converge after a finite number of iterations (Philpott and Guan (2008) Shapiro (2011)).

In the next section, we are going to investigate the DDP approach for the stochastic cases, which can deal with uncertain data and is more applicable to real-life situations.

2.6.2 Stochastic Dual Dynamic Programming

In the previous section, we have seen how DDP can solve a two-stage and then multistage DDP (deterministic) problems. In this section, we are going to investigate Stochastic Dual Dynamic Programming (SDDP). We are going to start with a two-stage stochastic problem and then extend it to a multistage stochastic problem.

A two-stage SDDP problem is given as:

$$\begin{aligned} \min \quad & {c_1}^T x_1 + \sum_{j=1}^{m} p_j {c_2}^T x_{2j} \\ \text{s.t.} \quad & A_1 x_1 \geq b_1 \\ & E_1 x_1 + A_2 x_{2j} \geq b_{2j}, j = 1, \ldots, m \end{aligned} \tag{2.34}$$

Where $c_1 \in \mathbb{R}^{n_1}$, $x_1 \in \mathbb{R}^{n_1}$, $p_j \in \mathbb{R}$, $c_2 \in \mathbb{R}^{n_2}$, $x_{2j} \in \mathbb{R}^{n_2}$, $A_1 \in \mathbb{R}^{m_1 \times n_1}$, $b_1 \in \mathbb{R}^{m_1}$, $E_1 \in \mathbb{R}^{m_2 \times n_1}$, $A_2 \in \mathbb{R}^{m_2 \times n_2}$, $b_{2j} \in \mathbb{R}^{m_2}$. Problem (2.34) can be interpreted as a two-stage problem with the first-stage decision x_1. Given the trial decision x_1, there are m second-stage problems (subproblems):

$$\begin{aligned} \alpha_{1j}(x_1) = \min \quad & {c_2}^T x_{2j} \\ \text{s.t.} \quad & A_2 x_{2j} \geq b_{2j} - E_1 x_1 \end{aligned} \tag{2.35}$$

Where $j = 1, \ldots, m$. Each subproblem is a scenario that occurs with the probability p_j. Let $\bar{\alpha}_1(x_1) = \sum_{j=1}^{m} p_j \alpha_{1j}(x_1)$. The first-stage problem in the DP recursion becomes:

$$\begin{aligned} \min \quad & {c_1}^T x_1 + \bar{\alpha}_1(x_1) \\ \text{s.t.} \quad & A_1 x_1 \geq b_1 \end{aligned} \tag{2.36}$$

All of the derivations with two-stage DDP problems can be applied to the stochastic case. Furthermore, it can be extended to the multistage stochastic program.

In the multistage stochastic problems, the SDDP algorithm performs one forward simulation and one backward simulation for every iteration. The purpose of the forward simulation is to find "good" trial decisions at each stage. Also, it gives the upper bound on the objective value of the problem. Then, given the trial decisions at each stage, the backward simulation will solve the subproblems corresponding to each scenario generated from the random variables. The dual obtained after solving these problems are used to construct the lower-bound piecewise-linear approximation of the future cost function at each stage. This technique is known as Bender's decomposition as explained in Section 2.1. Therefore, the objective value at the first stage

can also be used as the lower bound on the objective value of the whole problem.

Depending on different ways of upper bound, lower bound and confidence interval calculations, there are different ways to stop the SDDP algorithm (Shapiro (2011) Homem-de Mello et al. (2011) Homem-de Mello and Bayraksan (2014)). The stopping criterion we use throughout the thesis is given in Shapiro (2011) to avoid an early stop caused by the large variance of the upper bound estimator (Remark 4 in the paper Shapiro (2011)). Its principle is as follows: Every iteration of the SDDP algorithm includes an backward and an forward simulation. The backward simulation constructs the lower bound on the recourse function at every stage so the objective value at the first stage will also be the lower bound of the whole optimization problem. This is denoted as $\underline{\theta}$. Then, in the forward simulation, we have n paths from stage one to the final stage. For every projection j, we calculate the "true" cost:

$$\theta_i = \sum_{t=1}^{T} c_{ti}^T \hat{x}_{ti}, i = 1, \ldots, n \tag{2.37}$$

Where c_{ti} and $\hat{x}_{ti}$ are the cost and trial decicion vectors at stage t for the projection j. We then calculate the mean, $\bar{\theta}$, and the variance, $\hat{\sigma}_\theta^2$, of these paths:

$$\bar{\theta} = \frac{1}{n} \sum_{i=1}^{n} \theta_i \tag{2.38}$$

$$\hat{\sigma}_\theta^2 = \frac{1}{n-1} \sum_{i=1}^{n} (\theta_i - \bar{\theta})^2 \tag{2.39}$$

Then the confidence interval can be constructed as:

$$[\bar{\theta} - z_{\alpha/2}\hat{\sigma}_\theta/\sqrt{n}, \bar{\theta} + z_{\alpha/2}\hat{\sigma}_\theta/\sqrt{n}] \tag{2.40}$$

Where z_α denotes the $(1-\alpha)$-quantile of the standard Normal distribution. For example: $z_{0.025} = 1.96$ shows the 95% confidence interval. If the difference between the upper confidence bound $\bar{\theta} + z_\alpha \hat{\sigma}_\theta/\sqrt{n}$ and the lower bound $\underline{\theta}$ is less than a prescribed accuracy level $\epsilon > 0$, the algorithm stops. Typically, the value of ϵ is 10%, meaning that the upper bound is no more than 10% above the lower bound Homem-de Mello et al. (2011). With this stopping criterion, the optimization problem is guaranteed to be solved with accuracy ϵ for the $(1-\alpha)$ confidence.

The accuracy of SDDP and other sampling algorithms can be increased by increasing the number of samples n (Shapiro (2011) Homem-de Mello et al. (2011) Homem-de Mello and Bayraksan (2014)). However, increasing in the

number of samples may make the problem intractable or very computationally intensive. Our aim is to reduce the error while keeping a small number of samples. The multistage SDDP algorithm is summarized in Algorithm 2.

In the next section, we are going to show an example of how sampling algorithms can give wrong solutions when using a small number of samples. After that, we are going to review different Variance Reduction (VR) methods that can be used to solve this problem before proposing our own method based on the theory of Importance Sampling.

2.7 The perils of sampling in decomposition algorithms

In this section, we are going to show an example of how sampling algorithms can give wrong solutions when using a small number of samples. Firstly, we are going to introduce a newsvendor problem. Secondly, we are going to solve this problem using Crude Monte Carlo (CMC) method paired with a decomposition algorithm. The result shows that using a small number of samples may produce an invalid cut, leading to a high error in the recourse function's approximation. Even if the sampling error of every cut is small, it is compounded over a number of iterations. Eventually it can become so significant that it gives inaccurate solutions and objective value.

2.7.1 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 LP problem defined as,

$$\begin{aligned} z^* = \min_{x} \quad & x + \mathcal{Q}(x) \\ \text{s.t.} \quad & x \geq 0, \end{aligned} \tag{2.46}$$

where $\mathcal{Q}(x) = E_\xi\{Q(x,\xi)\}$. The second-stage function is a LP problem defined as,

$$\begin{aligned} Q(\widehat{x},\xi) = \min_{y_1,y_2} \quad & -p(\xi)y_1 - ry_2 \\ & y_1 \leq d(\xi), \\ & y_1 + y_2 \leq \widehat{x}, \\ & y_1,\ y_2 \geq 0, \end{aligned} \tag{2.47}$$

Algorithm 2 Multistage Stochastic Dual Dynamic Programming Algorithm

Step 1. Initialize: $\hat{\alpha}_t(x_t) = 0$ for $t = 1, \ldots, T$, where T is the total number of time periods (or stages); $\bar{z} = \infty$.
Step 2. Solve the approximate first-stage problem as in (2.30). Let $\hat{x}_1$ be the optimal solution.
Step 3. Calculate the lower bound $\underline{z}$ as in (2.32). If $\bar{z} - \underline{z} \leq \epsilon$, the problem is solved. Otherwise, go to step 4.
Step 4. For $t = 1, \ldots, T-1$ (forward simulation)
Sample n paths from stage one to the final stage. For $i = 1, \ldots, n$, solve the following optimization problem:

$$\begin{aligned} \hat{x}_{t+1,i} = \quad & \underset{}{\text{argmin}} \quad {c_{t+1}}^T x_{t+1,i} \\ & \text{s.t.} \qquad A_{t+1} x_{t+1,i} \geq b_{t+1,i} - E_t \hat{x}_t \end{aligned} \tag{2.41}$$

Store the optimal dual solution as $\pi_{t+1,i}$.
Step 5. Calculate the mean, $\bar{\theta}$, and the variance, $\hat{\sigma}_\theta^2$, of these paths:

$$\bar{\theta} = \frac{1}{n} \sum_{i=1}^{n} \theta_i \tag{2.42}$$

Where $\theta_i = \sum_{t=1}^{T} c_{ti}^T \hat{x}_{ti}$, and

$$\hat{\sigma}_\theta^2 = \frac{1}{n-1} \sum_{i=1}^{n} (\theta_i - \bar{\theta})^2 \tag{2.43}$$

Then construct the confidence interval:

$$[\bar{\theta} - z_{\alpha/2} \hat{\sigma}_\theta / \sqrt{n}, \bar{\theta} + z_{\alpha/2} \hat{\sigma}_\theta / \sqrt{n}] \tag{2.44}$$

Set $\bar{z} = \bar{\theta} + z_\alpha \hat{\sigma}_\theta / \sqrt{n}$
Step 6. For $t = T-1, \ldots, 2$ (backward recursion)
Solve the following optimization problem:

$$\begin{aligned} \min \quad & {c_t}^T x_t + \alpha \\ \text{s.t.} \quad & A_t x_t \geq b_t - E_{t-1} \hat{x}_{t-1} \\ & \alpha \geq \sum_{i=1}^{m} p_{t+1,i} \{\pi_{t+1,i}\}^T \{b_{t+1,i} - E_t \hat{x}_t\} \end{aligned} \tag{2.45}$$

Where the multipliers $\pi_{t+1,i}$ are obtained from the step 4.
Step 7. Return to step 2.

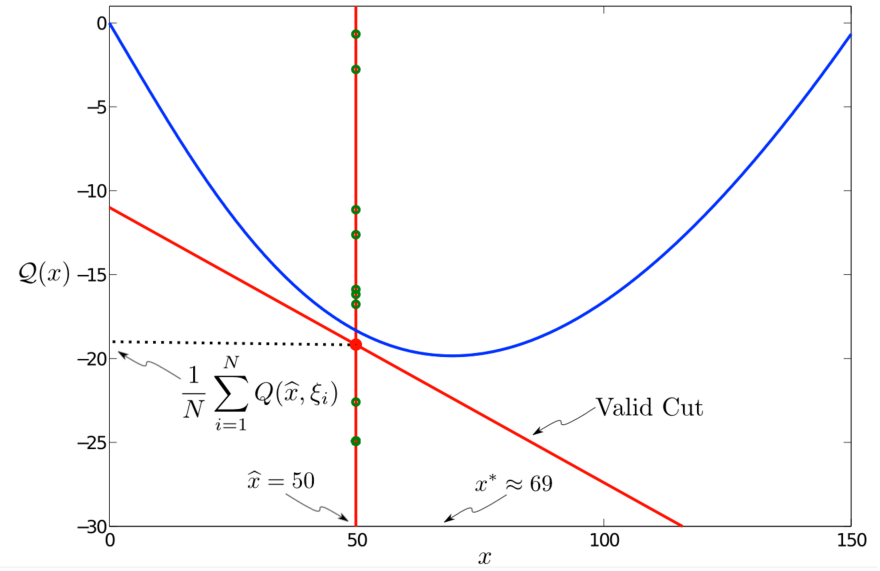
where $\widehat{x}$ denotes the quantity of newspapers purchased in the first stage, $\xi = (\xi_1, \xi_2)$ represents the uncertainty in demand $d(\xi)$ and sales price $p(\xi)$ of newspapers in the second-stage, and scalar r represents the price of recycling unsold newspapers. We usually model the 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 independent normal random variables with mean μ and and standard deviation σ. This implies that the uncertainty in $d(\xi)$ and $q(\xi)$ are modelled using a lognormal distribution.

2.7.2 Sampling error in decomposition algorithms

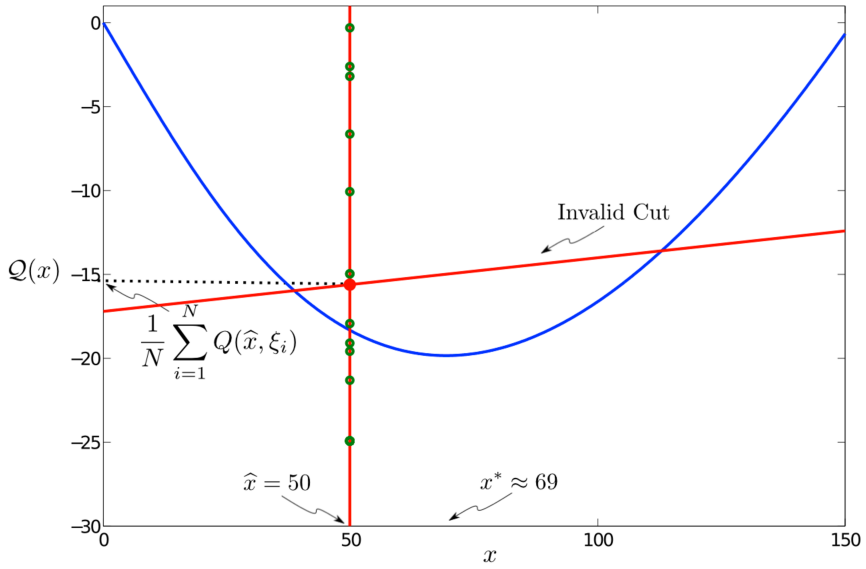
We solve the newsvendor problem (Section 2.7.1) using Crude Monte Carlo (CMC) method paired with a decomposition algorithm. Figure 2.1 shows an invalid cut for approximating the recourse function when there are only a small number of samples used in the decomposition algorithms.

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 2.1(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 2.1(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. However, it has other drawbacks. 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. 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 SP problems with a small number



(a)



(b)

Figure 2.1: In 2.1(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 2.1(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.

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.

It is well-known that we can reduce the sampling error in the cut parameters if we increase the number of samples. Even so, the $O(N^{-0.5})$ convergence rate of CMC 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 SP 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.

In the next section, we are going to review different Variance Reduction (VR) methods that can estimate the recourse function with a high precision (i.e.: small sampling error and variance) while using fewer samples than the CMC method. Moreover, the small variance of the estimator of the recourse function can lead to a fast convergence of the algorithm Homem-de Mello and Bayraksan (2014).

2.8 Variance-Reduction Methods

In this section, we are going to review different types of Variance Reduction (VR) methods. In the context of stochastic optimization algorithms, the VR methods are used to give a better estimator of the recourse function with a smaller variance and sampling error while using fewer samples than the CMC method. Some VR methods that are going to be studied are: Antithetic Variates (AV), Latin Hypercube Sampling (LHS), Quasi Monte Carlo (QMC) and Importance Sampling (IS).

2.8.1 Antithetic Variates

The Antithetic Variates (AV) method is a Variance Reduction method. Its principle is to generate negatively correlated pairs $(\underline{\xi}_j, \bar{\xi}_j), j = 1, \ldots, N/2$, such that $\underline{\xi}_j$ is generated from the Uniform distribution $U[0,1]^{d_\xi}$ and $\bar{\xi}_j$ is given as $1 - \underline{\xi}_j$. The AV estimator is then calculated as:

$$\frac{1}{N} \sum_{j=1}^{N/2} Q(x, \underline{\xi}_j) + Q(x, \bar{\xi}_j) \tag{2.48}$$

AV is an unbiased estimator. Its variance is given by,

$$\frac{\sigma^2(x)}{N} + \frac{\mathrm{Cov}(Q(x, \underline{\xi}_j), Q(x, \bar{\xi}_j))}{N} \tag{2.49}$$

According to Equation (2.49), the variance reduction of AV depends significantly on the term $\frac{\mathrm{Cov}(Q(x,\underline{\xi}_j),Q(x,\bar{\xi}_j))}{N}$. If this term is smaller than zero, the variance of the AV estimator is smaller than of the CMC method; otherwise, the variance of the AV estimator is increased relatively to the variance of the CMC estimator. In other words, AV becomes useful if and only if $Q(x, \underline{\xi}_j)$ and $Q(x, \bar{\xi}_j)$ have negative correlation. This property is shown to be satisfied in two-stage stochastic linear programs and when the uncertainty only occurs on the RHS (Higle (1998)). In some situations when the monotonicity of the objective function is lost, the variance of the AV method is larger than the variance of the CMC method (Koivu (2005)).

2.8.2 Latin Hypercube Sampling

Stratified Sampling reduces the variance of the estimator by splitting the sample space Ξ into K strata and then generating samples from every strata. The number of samples generated from each strata is proportional to the probability of that strata. Hence, the samples spread across the entire sample space. When $K = N$, it is known as the Latin Hypercube Sampling (LHS) (McKay et al. (1979)). In the two dimensional case, the sample space is considered as a square grid and LHS places one sample for each row and column. In the multidimensional case, LHS places only one sample in each axis-aligned hyperplane. It has been shown that the variance of the LHS estimator is always less than or equal to the variance of the CMC estimator (McKay et al. (1979)). In particular, their relationship is given as:

$$\mathrm{Var}_{\mathrm{LHS}} \leq \frac{N}{N-1}\mathrm{Var}_{\mathrm{CMC}} \tag{2.50}$$

Therefore, LHS always gives a better estimator than CMC.

LHS has also been widely used in stochastic optimization algorithms (Bailey et al. (1999), Shapiro et al. (2002), Linderoth et al. (2006), Homem-de Mello et al. (2011), Freimer et al. (2012)). It has been proven that the convergence rate of stochastic optimizations using LHS is never worse than those using CMC (Owen (1992) Drew and Homem-de Mello (2012)).

2.8.3 Quasi-Monte Carlo

Quasi-Monte Carlo (QMC) has been considered as one of the most popular VR methods (Niederreiter (1992),Lemieux (2009),Dick and Pillichshammer (2010)). The principle of QMC is that: instead of generating samples randomly from the Uniform distribution on $[0,1]^{d_\xi}$, the QMC samples are generated in a specific way that can achieve a high-quality estimator. The quality of an estimator, according to the Koksma-Hlawka inequality (Niederreiter (1992)), is determined by the quality of the generated samples such that the difference between the empirical distribution and the Uniform distribution is minimal. This property is known as "star-discrepancy". Moreover, the quality of an estimator depends on the total variation of the estimate function. There has been a large number of research papers on the construction of low-discrepancy sequences (Bastin et al. (2006) Freimer et al. (2012)). Some examples of such sequences are: Halton and Sobol' sequences.

In practice, the total variation of the estimate function can be very difficult to calculate. To overcome this problem, some randomness are introduced into the generated QMC samples so that the error can be estimated using the standard methods such as Multiple Independent Replications. Some examples of the randomized methods are: The Cranley-Patterson method and other scrambling algorithms (e.g.: Owen scrambling algorithm for Sobol sequences, Reverse-Radix 2 algorithm for Halton sequences). It has been shown that: The sampling error in the randomized QMC estimator converges at the rate $O(\frac{logN^{0.5*(d_\xi-1)}}{N^{1.5}})$ (Bastin et al. (2006) Freimer et al. (2012)). This error rate depends on the dimension of random variables, showing that QMC may converge slowly for high dimensional problems. To solve this problem, one can determine the *effective dimension* of the problem, and then apply QMC on these dimensions while applying CMC or LHS or some other efficient sampling methods on the other dimensions (Owen (1998) Owen (2003)).

In the context of stochastic optimization, QMC has been used for many years (Kalagnanam and Diwekar (1997) Drew and Homem-de Mello (2006)). The convergence of QMC in stochastic optimization has been studied by (Pennanen and Koivu (2005) Koivu (2005)). These papers show that the QMC can improve significantly the convergence rate of stochastic optimization algorithms.

2.8.4 Importance Sampling

Importance Sampling (IS) increases the quality of estimator by generating samples from a different distribution than the distribution of interest. The new distribution is called the *importance sampling* distribution. This is be-

cause the samples generated from this distribution are in the regions that contribute the most to the estimator. The principle of IS can be explained as follows: We want to calculate the expected value of a given function Q:

$$\mathbb{E}_f\{Q(\xi)\} = \int Q(\xi) f(\xi) d\xi \tag{2.51}$$

where $\xi \sim f$. Equation (2.51) is usually a very high dimensional integration so it is usually very difficult to evaluate. Using the Crude Monte Carlo (CMC) approach, equation (2.53) can be approximated as:

$$\mathbb{E}_f\{Q(\xi)\} = \frac{1}{n} \sum_{i=1}^{n} Q(\xi_i) \tag{2.52}$$

where $\xi_i \in \mathbb{R}^d$ are independent identical distributed (iid) samples and assumed to be drawn efficiently from the distribution f. Although CMC approximation is widely used in sampling methods due to its simple implementation, it has many drawbacks. Firstly, it is not always possible to draw samples efficiently from distribution f. Secondly, the variance of the estimator (2.52) is $\frac{\sigma^2(Q(\xi))}{n}$. This is usually large, making the error of our approximation large too. One possible solution is to take more samples in order to reduce the error. However, taking more samples means that the computation becomes more expensive or even intractable. It is therefore essential to find a way to reduce the variance while keeping the same number of samples. In other words, given the same number of samples, we want to find an alternative distribution, say g, that gives us a much lower variance and at the same time corrects for the fact that we are using distribution g, instead of using the original distribution f. The function g is called the IS distribution. The probability density function (pdf) of G is denoted as g. If we multiply and divide equation (2.51) by $g(\xi)$, the expected value remains the same:

$$E_f\{Q(\xi)\} = \int Q(\xi) f(\xi) d\xi = \int Q(\xi) \frac{f(\xi)}{g(\xi)} g(\xi) d\xi \tag{2.53}$$

with a condition that $g(\xi) = 0 \iff f(\xi) = 0$. Equation (2.53) shows that $E_f\{Q(\xi)\}$ can be approximated as:

$$E_f\{Q(\xi)\} \approx \frac{1}{n} \sum_{i=1}^{n} Q(\xi_i) \frac{f(\xi_i)}{g(\xi_i)} \tag{2.54}$$

where $\xi_i \sim g$. Define:

$$\lambda = \frac{f(\xi_i)}{g(\xi_i)} \tag{2.55}$$

λ is known as the likelihood ratio. This ratio is necessary for keeping the IS estimator unbiased.

The motivation for using Equation (2.54) instead of Equation (2.52) is that the variance of (2.52) is $\frac{1}{n}\sigma^2(Q(\xi))$ while the variance of (2.54) is $\frac{1}{n}\sigma^2(Q(\xi)\frac{f(\xi)}{g(\xi)})$. This suggests that: If we can select a good $g(x)$, we can achieve a great reduction in the variance. Choosing a good IS distribution, however, 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 IS.

The IS estimator (2.54) maintains all of the attractive properties of CMC estimator: It is an unbiased estimator; it is a consistent estimator i.e. the approximate value is getting closer to the true value with probability one as the number of samples n goes to infinity; the variance becomes zero as n goes to infinity; and the convergence rate is independent on the dimension of x.

IS has been used in stochastic optimization for a long time (Dantzig and Glynn (1990) Infanger (1992) Dantzig and Infanger (1993)). These papers have shown many advantages of IS when applied to stochastic optimization algorithms. For example: Large-scale multistage portfolio optimization problems can be solved efficiently (Dantzig and Infanger (1993)). Applying IS in stochastic optimization, in addition, helps capturing rare events (e.g.: power outage or a sudden rise in demand in the context of power generation and expansion planning for electric utilities) much more effective than the CMC method (Infanger (1992)).

2.9 Summary

Variance Reduction (VR) methods are techniques that are used to increase accuracy of the sampling algorithms. There are many different approaches such as Control Variates, Antithetic Sampling and Stratification. It is, however, not easy to apply these methods in SP algorithms, especially when the distribution of recourse function is dependent on the previous-stage solution. One promising VR method that can be used for solving SP problems is Importance Sampling (IS). IS helps sampling algorithms such as Bender's decomposition and SDDP for several reasons. Firstly, it requires a small number of samples to obtain a accurate estimator, and hence the number of LP problems that need to be solved is reduced. This can save a great amount of computational effort. Secondly, the accuracy of the estimate is increased by obtaining less variance. The variance affects the stopping criterion; therefore, reducing the variance also improves the convergence rate of the algorithm. Finally, it is easy to implement the IS algorithm if the IS

distribution is known.

Based on the theory of Importance Sampling, we are going to propose an algorithm to solve multistage SP problems efficiently and accurately in the next Chapter.

Chapter 3

Importance Sampling for Stochastic Programming algorithms

In multistage SP problems, the most computationally intensive part is the calculation of the recourse function $\mathbb{E}_f\{Q(\widehat{x}, \xi)\}$, where $\widehat{x}$ is the previous-stage decisions and $\xi \sim f$. Based on (2.54), the IS estimator for the recourse function is given as:

$$E_f\{Q(\widehat{x}, \xi)\} \approx \frac{1}{n} \sum_{i=1}^{n} Q(\widehat{x}, \xi_i) \frac{f(\xi_i)}{g(\xi_i)} \tag{3.1}$$

where $\xi_i \sim g$. IS is the most effective in the context of SP problems when g can generate samples from the regions that contribute the most to the value of the recourse function at a fixed point $\widehat{x}$. The variance of an IS estimator is minimized when we sample from the following IS distribution Asmussen and Glynn (2007):

$$g^*(\xi) = \frac{|Q(\widehat{x}, \xi)|}{\mathbb{E}_f|Q(\widehat{x}, \xi)|} f(\xi). \tag{3.2}$$

The IS distribution g^* is optimal in the sense that no other distribution can produce an IS estimator with smaller variance. 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 (3.2) in practice is that it requires us to know the value of $\mathbb{E}_f|Q(\widehat{x}, \xi)|$, 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 (3.2) to construct zero-variance estimates if we already have a zero-variance estimate of $\mathbb{E}_f|Q(\widehat{x}, \xi)|$.

The IS framework that we introduce in this thesis revolves around two key observations. The first observation is that we can generate samples from (3.2) using an Markov Chain Monte Carlo (MCMC) algorithm since we know the distribution up to a normalizing constant $\mathbb{E}_f|Q(\widehat{x},\xi)|$. 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 IS estimator because we need to evaluate the likelihood of each sample as shown in Equation (2.55). In this case, the likelihood of a given sample is given by,

$$\Lambda^*(\xi) = \frac{f(\xi)}{g^*(\xi)} = \frac{f(\xi)}{\frac{|Q(\widehat{x},\xi)|}{\mathbb{E}_f|Q(\widehat{x},\xi)|}f(\xi)} = \frac{\mathbb{E}_f|Q(\widehat{x},\xi)|}{|Q(\widehat{x},\xi)|}, \tag{3.3}$$

and it is also impossible to compute in practice as it depends on $\mathbb{E}_f|Q(\widehat{x},\xi)|$. This leads us to the second observation: while we cannot use the samples from (3.2) to directly form an IS estimator, we can use them to reconstruct an approximation of the zero-variance distribution using a Kernel Density Estimation (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 IS estimator. As a result, we propose a new sampling framework that 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

3. Sample from the approximate zero-variance distribution to form a lower-variance IS estimator.

This proposed sampling framework is the core of this thesis and we will explain its mechanism and applications in SP, as well as its benefits throughout this thesis.

In the next section, we will review some principles of MCMC and KDE algorithms, and then we will explain how they can be applied in SP.

3.1 Markov chain Monte Carlo: Sampling from the optimal IS distribution for the recourse function's approximation

Markov Chain Monte Carlo (MCMC) are methods that can be used to sample from a probability distribution. For every iteration, the chain generates a sample (i.e.: a state) such that after a number of iterations, these samples have an equilibrium distribution that is equivalent to the desired distribution. As a result, the quality of samples increases with the number of iterations. In addition, a good chain should have the *rapid mixing* property such that the target distribution can be achieved quickly given any arbitrary starting state.

In this thesis, we are going to use two types of MCMC methods: the Random Walk MCMC and the Adaptive Metropolis proposed in Haario et al. (2001). The Random Walk MCMC is a particular instance of the Metropolis-Hastings algorithm. The motivation for using Random Walk MCMC is that it is simple to implement and only requires the specification of the proposal distribution. The Adaptive Metropolis algorithm was proposed in Haario et al. (2001) and it increases the efficiency of the Metropolis-Hastings algorithm by constantly updating the proposal distribution using the full information accumulated so far. The advantages of using MCMC algorithm in SP algorithms will be demonstrated in Section 5.2, Section 5.3 and Section 6.3.

We will explain the principles of the Metropolis-Hastings algorithm in the next section.

3.1.1 Basic theory of Metropolis-Hastings algorithm

The Metropolis-Hastings algorithm was first introduced by Metropolis et al. (1953). After that, it was generalized by Hastings (1970). The algorithm has been used to solve many problems in image analysis, astronomy, genetics and so on Gilks et al. (1996). One of its main advantages is that it can draw samples from difficult multivariate distributions for which direct sampling is not easy to be done (e.g.: when the normalization factor of the target distribution is difficult to obtain in practice). Other advantages are the generated samples can be used to approximate the distribution (i.e. construct the density estimation of the generated samples) or to perform a numerical integration. The principles of Metropolis-Hastings algorithm are as follows: It generates a Markov chain whose distribution after a sufficient long run matches the target distribution. Let $g(x)$ be the target probability

density function (pdf). The algorithm first starts at a random state. It then randomly moves to different points in the sample space. At each point, it performs a test whether to accept or reject the proposed move. If the test shows that the move goes from low to high-density region, we always accept the move. On the other hand, if the test shows that the move goes from high to low-density region, we reject the move with a certain probability. The probability of rejection depends on how much the density will decrease as the sample moves from the current position to the proposal position. By performing the test, the algorithm ensures that most of the generated samples stay in the high-density region and only a few generate samples stay in the low-density region. The full explanation of the algorithm can be found in Chib and Greenberg (1995).

In terms of mathematical framework, it initially starts at an arbitrary position, say $\xi_0 \in \mathbb{R}^d$, as the first sample. The next sample $\zeta_0 \in \mathbb{R}^d$ is proposed according to the *proposal distribution* $q(\zeta_0|\xi_0)$. As we generate more samples, the proposal distribution is denoted as $q(\zeta_k|\xi_k)$, where ξ_k is the current sample at the k-th iteration and ζ_k is the *proposal* sample. A commonly used proposal distribution is the Gaussian distribution centred at the current sample ξ_k. The proposal sample ζ_k is then accepted with a probability:

$$a(\xi_k, \zeta_k) = \frac{g(\zeta_k)q(\xi_k|\zeta_k)}{g(\xi_k)q(\zeta_k|\xi_k)} \tag{3.4}$$

where $a(\xi_k, \zeta_k)$ is known as the *acceptance ratio*. If $a(\xi_k, \zeta_k) \geq 1$, ζ_k is obviously accepted; and hence $\xi_{k+1} = \zeta_k$. If $a(\xi_k, \zeta_k) < 1$, we generate a random variable α from the uniform distribution $U(0, 1)$. If $\alpha \leq a(\xi_k, \zeta_k)$, ζ_k is accepted i.e. set $\xi_{k+1} = \zeta_k$; otherwise, ζ_k is rejected i.e. set $\xi_{k+1} = \xi_k$.

The advantages of Metropolis-Hastings algorithm are: It is easy to implement, it does not require the specification of many parameters, and it does not depend on a restrictive set of assumptions. We refer the interested reader to Gelman et al. (2010) for more information on the Metropolis-Hastings algorithm, and other MCMC algorithms that can be used in our proposed IS framework.

In the next section, we will explain how the Metropolis-Hastings algorithm can be used to generate a finite number of samples according to the optimal IS distribution (3.2) in order to achieve a good approximation of the recourse function in a SP problem.

3.1.2 Metropolis-Hastings for Stochastic Programming algorithms

As mentioned before, the Metropolis-Hastings algorithm generates a Markov chain whose distribution after a sufficient long run matches the target distribution. In a SP problem, we want this target distribution to be as close as possible to the IS distribution, g^*, as defined in Equation (3.2).

Let the sample at the k-th iteration be $\xi_k \in \mathbb{R}^d$. Then, a proposal sample $\zeta_k \in \mathbb{R}^d$ is suggested with the proposal distribution $q(\zeta_k|\xi_k)$. For the Random Walk MCMC, the proposal sample is given as:

$$\zeta_k = \xi_k + v_k \tag{3.5}$$

where v_k is the multivariate Normal distribution with mean 0 and covariance matrix Σ. The proposal sample is accepted i.e. $\xi_{k+1} = \zeta_k$ with the probability:

$$a(\xi_k, \zeta_k) = \min\left\{\frac{|Q(\hat{x}, \zeta_k)|f(\zeta_k)q(\xi_k|\zeta_k)}{|Q(\hat{x}, \xi_k)|f(\xi_k)q(\zeta_k|\xi_k)}, 1\right\} \tag{3.6}$$

where $a(\xi_k, \zeta_k)$ is known as the acceptance ratio; $Q(\hat{x}, \zeta_k)$ is the future cost function of the current decision variable $\hat{x}$ and random variable ζ_k. If $a(\xi_k, \zeta_k) \geq 1$, ζ_k is obviously accepted; and hence $\xi_{k+1} = \zeta_k$. If $a(\xi_k, \zeta_k) < 1$, we generate a random variable α from the uniform distribution $U(0, 1)$. If $\alpha < a(\xi_k, \zeta_k)$, ζ_k is accepted i.e. set $\xi_{k+1} = \zeta_k$; otherwise, Z is rejected i.e. set $\xi_{k+1} = \xi_k$.

The Metropolis-Hastings algorithm in SP is summarized in Algorithm 3. Algorithm 3 runs until we accumulate the required number of samples M. In this thesis, the value of M is selected based on our numerical experiments in Chapter 5.

The next step in our proposed IS algorithm is to construct the optimal (zero-variance) IS distribution i.e.: construct g^*. In one dimension, g^* can be estimated using a histogram. However, histograms cannot be applied to the multidimensional case. There are many requirements that need to specify in order to construct a multivariate histogram: the bin origin, the bin size and the bin orientation. The bin size normally has to be relatively small in order to capture reasonably local information, resulting in a large number of bins. In more than two dimensions, the number of bins quickly gets out of hand and the random effects are likely to become dominant. Furthermore, it is not easy to grasp the structure of data with a multivariate histogram. This is due to the fact that histogram is made of discontinuous bins, not a smooth function. One solution to overcome these problems is to use Kernel Density Estimation (KDE), which will be explained in the next section. For

Algorithm 3 Metropolis-Hastings Algorithm

Given the current state Y_k

1. Generate Z with the proposal distribution $q \sim N(Y_k, \sigma^2 I)$
2. Generate a random variable α from the uniform distribution $U(0,1)$
3. Compute the next state of the Markov chain:

$$Y_{k+1} = \begin{cases} Z & \text{if } \alpha < a(Y_k, Z) \\ Y_k & \text{otherwise} \end{cases}$$

where,

$$a(Y_k, Z) = \min\left\{\frac{|Q(\hat{x}, Z)f(Z)|}{|Q(\hat{x}, Y_k)f(Y_k)|}, 1\right\}$$

the complete literature review on KDE, please refer to (Silverman (1986) Scott (1992)).

3.2 Kernel Density Estimation: 1/ Constructing the optimal Importance Sampling distribution 2/ Correcting the bias in the estimator

A quick summary of the work described so far: Samples are generated according to the optimal IS distribution (3.2) such that most of the samples stay in the region that makes the most contribution to the calculation of the recourse function. However, the estimator using this optimal IS distribution is biased. In order to correct this bias, we need to calculate the so-called *likelihood ratio*. We propose to use the Kernel Density Estimation (KDE) to estimate the optimal IS density. Another advantage for using KDE to construct the optimal IS distribution is that: We can generate more samples with a very small computational cost while these samples can increase significantly the accuracy of the estimator.

In the next section, we will explain the basic theory of KDE, and then how to apply it in SP algorithms.

3.2.1 Basic theory of Kernel Density Estimation

By definition, the probability density of a univariate random variable $\xi \in R$ is given as:

$$f(\xi) = \lim_{h \to 0} \frac{1}{h} P(\xi - \frac{h}{2} < \xi < \xi + \frac{h}{2}) \tag{3.7}$$

For any given $h \in R$, we can estimate $P(\xi - \frac{h}{2} < \xi < \xi + \frac{h}{2})$ by counting the number of samples falling in the interval $(\xi - \frac{h}{2}, \xi + \frac{h}{2})$. Thus the pdf of ξ can be estimated as:

$$\widehat{f}(\xi) = \frac{k}{nh} \tag{3.8}$$

where k is the number of samples falling in the interval $(\xi - \frac{h}{2}, \xi + \frac{h}{2})$; h is the length of the interval; n is the total number of samples. Equation (3.8), therefore, can be rewritten as:

$$\widehat{f}(\xi) = \frac{1}{nh} \sum_{i=1}^{n} w(\frac{\xi - \xi_i}{h}) \tag{3.9}$$

where $w(.)$ is the *weight function* defined as:

$$w(x) = \begin{cases} 1 & \text{if } |x| < \frac{1}{2}, \\ 0 & \text{otherwise} \end{cases} \tag{3.10}$$

The *naive estimator* is very similar to the histogram formulation in which the probability density of ξ is estimated as:

$$\widehat{f}(\xi) = \frac{\text{Number of samples in the same bin as } \xi}{\text{Bin width } \times \text{ The total number of samples}}$$

The difference between the *naive estimator* and histogram is that: the *naive estimator* does not require the specification of bin locations. In the *naive estimator*, each sample is used as the centre of bin. The *naive estimator*, however, still has many drawbacks. Firstly, the density function is not a continuous function and therefore may cause some problems with interpreting the data structure, especially in multivariate cases. Secondly, the *naive estimator* weights all of the samples ξ_i equally, regardless of their distance to the estimator point ξ. To avoid these problems, we use Kernel Density Estimation (KDE), which is a generalization of the *naive estimator*. In particular, KDE replaces the *weight function* described in Equation (3.10) with a smooth *kernel function*, which satisfies the following condition:

$$\int_{-\infty}^{\infty} K(\xi) d\xi = 1 \tag{3.11}$$

$K(.)$ is usually, but not always, a symmetric probability density function (for example: the Normal density) that satisfies the following conditions:

$$\int_{-\infty}^{\infty} tK(t)dt = 0 \tag{3.12}$$

$$\int_{-\infty}^{\infty} t^2K(t)dt = k_2 \neq 0 \tag{3.13}$$

One popular choice of $K(.)$ is the Gaussian kernel function:

$$K(x) = \frac{1}{\sqrt{2\pi}} \exp\{-\frac{1}{2}x^2\} \tag{3.14}$$

The KDE is then calculated as:

$$\widehat{f}(\xi) = \frac{1}{nh} \sum_{i=1}^{n} K(\frac{\xi - \xi_i}{h}) \tag{3.15}$$

where h is called the *bandwidth* or *smoothing* parameter. Different types of kernel functions are shown in Figure 3.1. Just as the *naive estimator* can be constructed as the sum of "boxes" placed on each sample, the kernel estimator is the sum of "bumps" placed on each sample as shown in Figure 3.2. The shape of "bumps" are determined by the type of kernel function K, while their width are determined by the smoothing parameter h. It is essential to select a good smoothing parameter as it makes a significant impact on the quality of the density estimator. A too large smoothing parameter may oversmooth the density estimator. Then we may lose out some important information. On the other hand, a too small smoothing parameter may cause the density estimator to be very spiky. In this case, it may be very difficult to interpret the data. Figure 3.3 demonstrates the effects of varying smoothing parameter on the quality of the estimator. Therefore, the optimal value of h is selected in such a way that it can minimize the error between the estimated and the true density.

When considering the density estimation at a single point, a natural measure is the *mean square error* (MSE), given as:

$$\begin{aligned} MSE_{\xi}(\widehat{f}) &= E\{\widehat{f}(\xi) - f(\xi)\}^2 \\ &= \left\{E\{\widehat{f}(\xi)\} - f(\xi)\right\}^2 + \mathrm{Var}\widehat{f}(\xi) \\ &= \text{Bias of the estimator}^2 + \text{Variance of the estimator} \end{aligned} \tag{3.16}$$

Equation (3.16) shows the trade-off between the bias and variance: For a given MSE, the bias is only decreased if the variance is increased, and vice

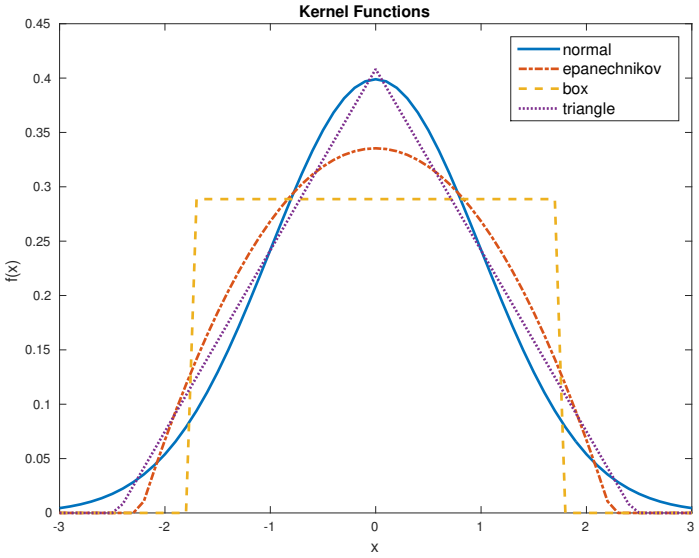


Figure 3.1: Different types of kernel functions used to calculate the density estimation

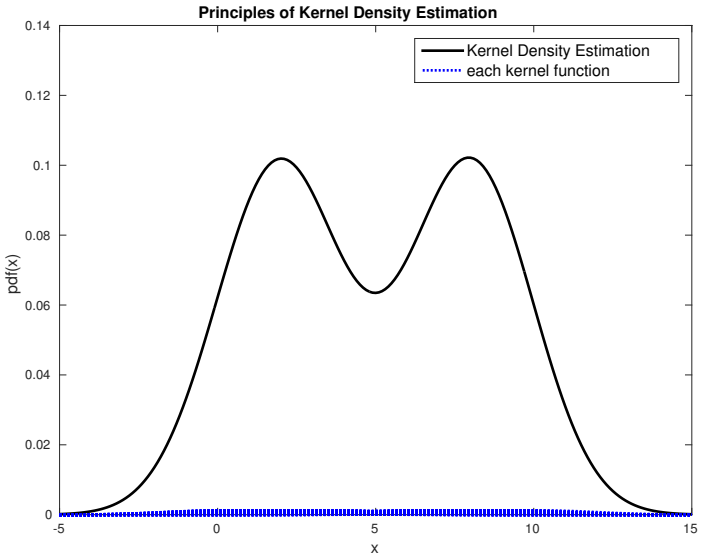


Figure 3.2: Principles of Kernel Density Estimation. Data are drawn from the Bimodal distribution which is the mixture of two Normal distributions. They are $N(2,1)$ and $N(8,1)$.

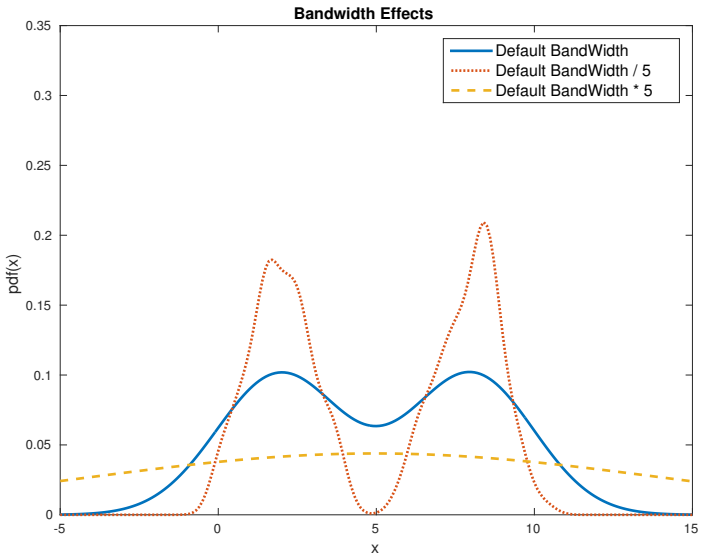


Figure 3.3: The effects of varied bandwidth on the quality of Kernel Density Estimation. Data are drawn from the Bimodal distribution which is the mixture of two Normal distributions. They are $N(2,1)$ and $N(8,1)$.

versa, by adjusting the amount of smoothing. This result is illustrated in Figure 3.4 and Figure 3.5. For a large smoothing parameter (Figure 3.4), it makes little difference among the estimates for different data sets, and hence obtains a low variance. However, they have a large bias since their shapes do not follow the true density. The opposite phenomenon occurs when we use a small smoothing parameter (Figure 3.5) i.e.: the kernel density estimation is able to show similar shapes between the estimated and true density; however, there is a large variance among the estimations for different sets of samples.

We can globally measure the accuracy of the density estimator with respect to the true density by taking the integration of equation (3.16) over all ξ. This quantity is known as the *mean integrated square error* (MISE):

$$MISE(\widehat{f}) = \int_{-\infty}^{\infty} \Big\{ E\{\widehat{f}(\xi)\} - f(\xi) \Big\}^2 d\xi + \int_{-\infty}^{\infty} \mathrm{Var}\widehat{f}(\xi) d\xi \quad (3.17)$$

According to Silverman (1986), MISE can be simplified as:

$$MISE(\widehat{f}) = \frac{1}{4} h^4 {k_2}^2 \int_{-\infty}^{\infty} f''(\xi)^2 d\xi + \frac{1}{nh} \int_{-\infty}^{\infty} K(t)^2 dt \quad (3.18)$$

where k_2 is defined in Equation (3.13). Since the *smoothing* parameter h determines significantly the quality of the density estimator, it should be

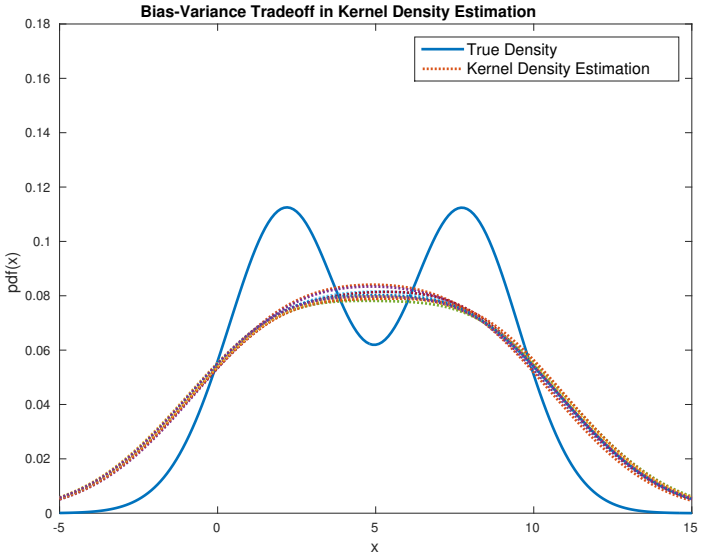


Figure 3.4: Trade-off between the bias and variance in the Kernel Density Estimation. When the bandwidth is large, the variance of estimations is small; The bias is large since their shapes do not follow the true density.

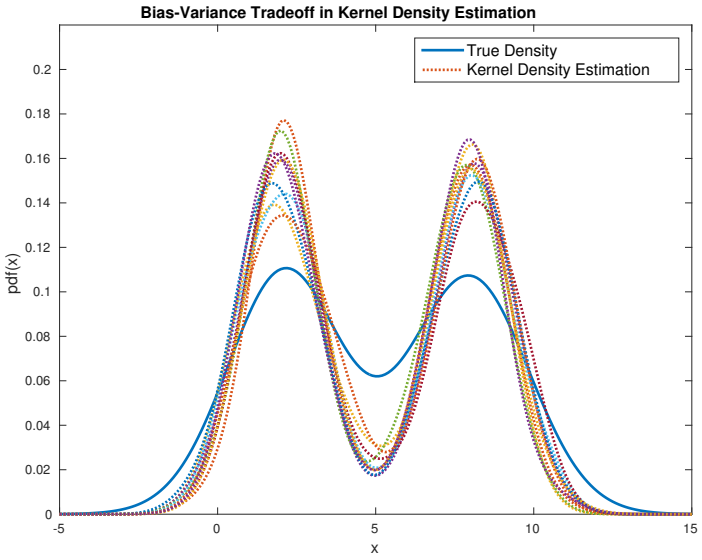


Figure 3.5: Trade-off between the bias and variance in the Kernel Density Estimation. When the bandwidth is small, the bias of estimations is small because their shapes follow the true density; the variance, however, is large.

chosen carefully to minimize the error between the density estimator and the true density. By minimizing the MISE with respect to h, the optimal value of h is given as Parzen (1962):

$$\begin{aligned} h_{opt} &= \arg\min_{h}\{MISE(\widehat{f})\} \\ &= {k_2}^{-2/5}\Big\{\int_{-\infty}^{\infty} f''(\xi)^2 d\xi\Big\}^{-1/5} n^{-1/5} \end{aligned} \tag{3.19}$$

This is not implementable in practice as h_{opt} still depends on the unknown value of $f(x)$. However, if we assume that the true distribution is Gaussian and the kernel function is the Gaussian kernel, Equation (3.19) can be derived as:

$$h_{opt} = 1.06 * \sigma * n^{-1/5} \tag{3.20}$$

where σ is the sample standard deviation and n is the total number of samples. Therefore, a quick way of selecting the *smoothing* parameter is by calculating σ from the data and substituting it into equation 3.20. A better result can be obtained by using a more robust measure of spread. The robust measure of spread can be the Inter-Quartile Range (IQR), which is the difference between $75th$ percentile (Q3) and the $25th$ percentile (Q1). Also, it is shown in Silverman (1986) that the coefficient 1.06 should be decreased to better cope with multimodal density. The optimal *smoothing* parameter therefore becomes:

$$h_{opt} = 0.9An^{-1/5} \tag{3.21}$$

Where $A = \min(\sigma, \frac{IQR}{1.34})$. The *smoothing* parameter selected in Equation (3.21) works well for a wide range of density estimation problems. For many purposes, it is certainly an adequate choice of *smoothing* parameter, for the others it is a good starting point for some subsequent fine tuning (Scott (2015)). There are many more sophisticated methods for selecting a good smoothing parameter, such as the least-square cross-validation, maximum-likelihood cross-validation. For a brief review on these literatures, please refer to (Silverman (1986)) and (Jones et al. (1996)).

In addition, based on the MISE, Hodges and Lehmann (1956) suggested a method for selecting the kernel function. The idea is to calculate the *efficiency* of different symmetric kernels K and compare them with the standard efficiency of Epanechnikov kernel. It is surprising that all of the kernels have the efficiencies very close to one and there is not much difference among them. This is consistent with the numerical results shown in Figure 3.6. The selection of kernel function is therefore based on other considerations such as the computational aspects.

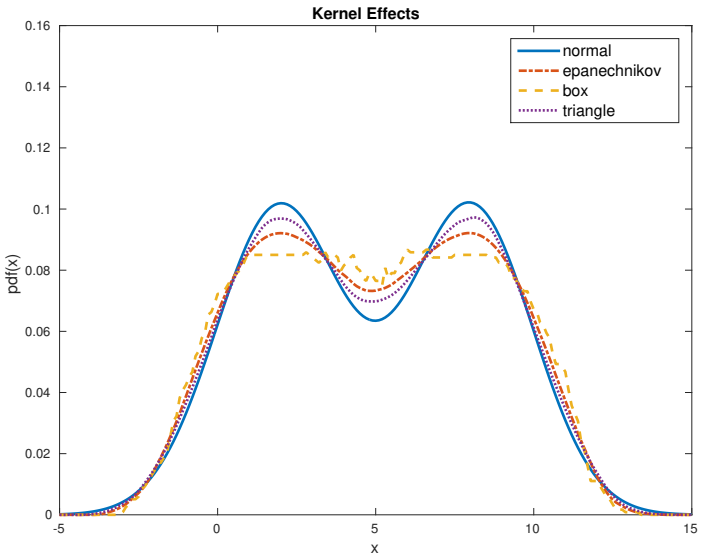


Figure 3.6: Different types of kernel functions can be used in the Kernel Density Estimation. There are little differences in the estimations.

All of the formulas and attractive features that we have learned about the univariate KDE can be applied directly for the multivariate KDE. In the multivariate cases, the arguments for using the KDE rather than histogram becomes much stronger. Firstly, it is much easier to present and grasp the structure of multidimensional density function in a continuous form than in discontinuous form. Secondly, it does not require the specification of bin locations and bin orientation as kernels are placed on each sample. The number of kernels is just equal to the number of samples.

The multivariate KDE is given as:

$$\widehat{f}(x) = \frac{1}{n}\sum_{i=1}^{n} K(\xi, \xi_i) \tag{3.22}$$

where

$$K(\xi, \xi_i) = \prod_{k=1}^{d} \frac{1}{h_k} K\{\frac{\xi_k - \xi_{i,k}}{h_k}\} \tag{3.23}$$

According to (3.23), the kernel function now consists of the product of d one-dimensional kernels. The same kernel function is applied to every dimension. The smoothing parameter, however, can be varied according to the spread of data in each dimension. The smoothing parameter for each dimension, hence, can be determined by any methods used in the univariate case.

In summary, we have reviewed the theory of Kernel Density Estimation. In the next section, we will explain how it can be used in SP algorithms.

3.2.2 Kernel Density Estimation in SP algorithms

In this section, we are going to apply KDE in SP algorithms. The basic idea is as follows: when solving SP problems, we have a set of samples that are used to approximate the recourse function; we can increase the accuracy of the approximation by either increasing the number of samples or decreasing the variance of the estimator. Since generating more samples is, in general, very expensive, our aim is to reduce the variance of the estimator. Based on the theory of importance sampling, a significant variance reduction can be achieved if samples are generated with the distribution proportional to the future cost function $|Q(x,\xi)|f(\xi)$. We have shown that these samples can be obtained effectively by Markov chain Monte Carlo algorithms (Section 3). However, if we apply Monte-Carlo approximation techniques directly on these samples, we will obtain a biased estimator. This is because the estimator is currently performed under the IS dsitribution but not under the true underlying distribution. In order to correct this bias, we have to multiply the biased estimator with the likelihood ratio. The likelihood ratio measures the proportion of the original distribution with respect to the IS distribution, and therefore can tell you how much to get from the biased estimator (estimator under the "biased" importance sampling distribution) to an unbiased estimator (estimator under the original "true" distribution). The IS distribution can be estimated by KDE. We can generate more samples from this IS distribution with little marginal computational cost while the extra samples can increase the accuracy of the estimator.

In Section 3.1.2, we show that a set of M samples can be generated from the optimal IS distribution using an MCMC algorithm. Using KDE, the optimal IS distribution can be estimated as:

$$\widehat{g}(\xi) = \frac{1}{M}\sum_{i=1}^{M} K_H(\xi, \xi_i) \tag{3.24}$$

where $K_H(.,.)$ is a kernel function satisfying the conditions of 3.11, 3.13 and $H \in \mathbb{R}^{D\times D}$ is its associated bandwidth matrix.

$K_H(.,.)$ is normally a Gaussian kernel, defined as:

$$K_H(\xi, \xi_i) = \prod_{k=1}^{d} \frac{1}{\sqrt{2\pi}h_k} \exp\{-\frac{\|\xi_k - \xi_{i,k}\|^2}{2{h_k}^2}\} \tag{3.25}$$

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.

The idea behind the one-dimensional likelihood-based search is as follows: For each dimension k, we define the function, $\hat{g}_{-i}(\xi)$, which is the density estimation constructed by all the data points except for ξ_i:

$$\hat{g}_{-i}(\xi) = \frac{1}{(n-1)h} \sum_{j \neq i} K(\frac{\xi - \xi_j}{h}) \tag{3.26}$$

The log likelihood function is defined as:

$$\mathrm{CV}(h) = \frac{1}{M} \sum_{i=1}^{M} \log \hat{g}_{-i}(\xi_i) \tag{3.27}$$

Then we find h to maximize the log likelihood function $\mathrm{CV}(h)$. This is the optimal value for h because it maximizes the probablity that all of the data points $\xi_1, \ldots, \xi_M$ are drawn from the density function $\hat{g}$. There has been a huge interest in selecting the optimal bandwidth for KDE and how it can be tailored for dependent data. Most of them show that there is not much difference in the smoothing parameter for dependent and independent data, unless the variables are extremely long-range and strongly dependent.

Using the approximate zero-variance distribution $\widehat{g}(\xi)$, we can finally construct an IS estimator 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.

Because the proposed algorithm uses Markov Chain Monte Carlo and Importance Sampling, we call it MCMC-IS. The full MCMC-IS algorithm is described in Algorithm 4. In the next Chapter, we are going to prove

the convergence of the algorithm. After that, we will test the proposed algorithm on a large number of applications as described in Ariyawansa and Felt (2004) in Section 5.1. Then, we will use the proposed algorithm to solve the capacity expansion planning in the electric power industry, which is a very large scale optimization problem integrating the unit commitment model and maintenance scheduling model, in Section 5.3.

Algorithm 4 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 from the optimal IS distribution;
Require: ξ_0: starting state of the MCMC-IS algorithm

`Step 1: Generate Samples from the optimal IS 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,

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

where,

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

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 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`

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

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

Chapter 4

Convergence of the MCMC-IS algorithm

In this chapter, we are going to prove the convergence of the proposed MCMC-IS algorithm. The convergence of MCMC, KDE and SDDP have been well studied individually (Robert and Casella (2013) Gelman et al. (2010) Haario et al. (2001) Silverman (1998) Athreya and Atuncar (1998) Shapiro (2011) Philpott and Guan (2008)). Therefore, the main task when trying to prove the convergence of the proposed MCMC-IS algorithm is to check the conditions for the convergence are satisfied. This chapter is organized into three sections. The first section establishes the convergence of MCMC in the MCMC-IS algorithm. The second section shows the convergence of KDE for a Markov chain that has been generated. The third section establishes the convergence of SDDP when using the proposed sampling framework for multistage SP problems.

4.1 Convergence of Markov Chain Monte Carlo in the MCMC-IS algorithm

In this section, we are going to show the convergence of Metropolis-Hastings when it is used in a Stochastic Programming algorithm as proposed in Section 3.1.2.

To show that a Markov chain converges, we have to prove that it has a stationary distribution, as well as it has the *aperiodic* and *Harris recurrent* properties. It has been shown that the Detailed Balance Condition described in the Definition 4.1 is the necessary and sufficient condition for a Markov chain to have a stationary distribution (Robert and Casella (2013)). For the completion of the thesis, we include the proof in Theorem 4.1. We

then show that the Metropolis-Hastings (MH) algorithm developed for the Stochastic Programming (SP) algorithm is based on the same principle of the conventional MH method. Therefore, based on the convergence proof for the conventional MH method, we show that our proposed MH method for SP algorithms will also converge to a stationary distribution, and this distribution is the optimal distribution (known up to a normalizing constant) for the calculation of the recourse function in SP problems according to Equation 3.2.

Definition 4.1. *(Robert and Casella (2013))* ***(Detailed Balance Condition)*** *A Markov chain satisfies the* detailed balance condition *if there exists a transition kernel K and a function g such that*

$$K(\zeta_k, \xi_k)g(\zeta_k) = K(\xi_k, \zeta_k)g(\xi_k) \tag{4.1}$$

for every (ξ_k, ζ_k)

Theorem 4.1. *(Robert and Casella (2013))* ***(Stationary distribution)*** *Suppose that a Markov chain satisfies the detailed balance condition with a transition kernel K and a function g. Then g is the invariant density of the chain*

Proof. (Robert and Casella (2013)) Because the Markov chain satisfies the detailed balance condition,

$$\begin{aligned} \int_{\zeta_\mathbf{k}} K(\zeta_k, B)g(\zeta_k)d\zeta_k &= \int_{\zeta_\mathbf{k}} \int_B K(\zeta_k, \xi_k)g(\zeta_k)d\xi_k d\zeta_k \\ &= \int_{\zeta_\mathbf{k}} \int_B K(\xi_k, \zeta_k)g(\xi_k)d\xi_k d\zeta_k \\ &= \int_B g(\xi_k)d\xi_k \end{aligned} \tag{4.2}$$

for any measurable set B. Note that: $\int_{\zeta_k} K(\xi_k, \zeta_k)d\zeta_k = 1$. By the Definition 4.5.1 in Robert and Casella (2013), g is the invariant density of the chain. The detailed balance condition and the reversibility of a chain are equivalent when there exist a transition kernel K and an invariance density g. □

Using the Detailed Balance Condition in Definition 4.1 and Theorem 4.1, Robert and Casella (2013) proved the convergence of the MH algorithm in Theorem 4.2.

Theorem 4.2. *(Robert and Casella (2013)) For every proposal distribution q, which has the support Ψ, g is the stationary distribution of the chain produced by the Metropolis-Hastings.*

Proof. Robert and Casella (2013) The transition kernel K given by the Metropolis-Hastings algorithm is:

$$K(\xi_k, \zeta_k) = q(\zeta_k|\xi_k)a(\xi_k, \zeta_k) + (1 - r(\xi_k))\delta_{\xi_k}(\zeta_k) \tag{4.3}$$

where q is the proposal distribution, $a(\xi_k, \zeta_k) = \min\{\frac{q(\xi_k|\zeta_k)g(\zeta_k)}{q(\zeta_k|\xi_k)g(\xi_k)}, 1\}$; $r(\xi_k) = \int q(\zeta_k|\xi_k)a(\xi_k, \zeta_k)d\zeta_k$; δ_{ξ_k} denotes the Dirac mass in ξ_k. The acceptance-rejection process in the Metropolis-Hastings algorithm can be written in another mathematical forms as follows:

$$q(\zeta_k|\xi_k)a(\xi_k, \zeta_k)g(\xi_k) = q(\xi_k|\zeta_k)a(\zeta_k, \xi_k)g(\zeta_k) \tag{4.4}$$

And

$$(1 - r(\xi_k))\delta_{\xi_k}(\zeta_k)g(\xi_k) = (1 - r(\zeta_k))\delta_{\zeta_k}(\xi_k)g(\zeta_k) \tag{4.5}$$

By adding Equation 4.4 and 4.5, we obtain the detailed balance condition of the Metropolis-Hastings chain. By Theorem 4.1, g is the stationary (invariant) distribution of the Metropolis-Hastings chain. □

Our proposed MH method for SP algorithms uses a different acceptance ratio to the conventional method such that the target stationary distribution should be $|Q(\hat{x}, \xi)|f(\xi)$ (according to Equation 3.2). As a result, the acceptance ratio in our proposed algorithm is: $a(\xi_k, \zeta_k) = \min\{\frac{q(\xi_k|\zeta_k)|Q(\hat{x},\zeta_k)|f(\zeta_k)}{q(\zeta_k|\xi_k)|Q(\hat{x},\xi_k)|f(\xi_k)}, 1\}$. The next step is to prove that the Markov chain generated by our method is aperiodic and Harris recurrent. The definitions of *irreducibility* and *Harris recurrence* are given in Definition 4.2 and Definition 4.3.

Definition 4.2. *(Robert and Casella (2013))* ***(Irreducibility)*** *Given a measure g, the Markov chain with transition kernel $K(\xi_k, \zeta_k)$ is g-irreducible if, for every set A with $g(A) > 0$, there exists n such that $K^n(\xi_k, A) > 0$ for all $\xi_k \in \Xi$. The chain is strongly g-irreducible if $n = 1$ for all measurable A.*

Definition 4.3. *(Roberts and Rosenthal (2006))* ***(Harris recurrence)*** *A Markov chain X is Harris recurrent if there exists a σ-finite measure g on the state space S such that for any set A where $g(A) > 0$, we have $P_x(\tau_A < \infty) = 1, \forall x \in S$, where τ_A is the hitting time to the set A and $P_x(\tau_A < \infty)$ is the probability of hitting set A in a finite amount of time given that the starting point is x.*

In other words, a Markov chain is *aperiodic* when the states of its chain do not repeat in a certain manner. This property is satisfied by the fact that the MH method developed for SP algorithms uses a Random Walk proposal mechanism. Hence, given the current state, the next state can be anywhere in the state space, which makes the sampling process completely random.

In addition, a Markov chain is *recurrent* if it is *irreducible* (i.e.: all of the states connects to each other) and any state can be visited by an infinite number of times. The *Harris recurrence* property is stronger than the *recurrence* property such that all of the generated Markov chains are recurrent (i.e.: for any Markov chain generation, each state can be visited by an infinite number of times). The *Harris recurrence* property is important because it shows that a Markov chain can converge to a stationary distribution regardless of its initial state. The *irreducibility* of our proposed MH method for SP algorithms can be verified by the fact that our method uses the same Random Walk proposal mechanism as in the conventional MH method, such that $q(\zeta_k|\xi_k) > 0$ for every ζ_k and ξ_k. In other words, given that the current state is ξ_k, it is always possible for the chain to move to the state ζ_k. As a result, the Markov chain generated by our proposed method is recurrent. In order to show the Harris recurrence property for our MH method, we show that our method is just an extension of the conventional MH method, which has been proven to be Harris recurrent by Tierney (1994) based on the theory of harmonic functions. The theory states that: A Markov chain is Harris recurrent if the only bounded *harmonic functions*[1] are constant (Tierney (1994)). The proof for the Harris recurrence property of the conventional MH method is shown in Lemma 4.1.

Lemma 4.1. *Tierney (1994) If the Metropolis-Hastings chain is irreducible with a stationary distribution g, it is Harris recurrent.*

Proof. Consider the Metropolis-Hastings chain,

$$\mathbb{E}_g[h(\xi_2)|\xi_1] = \int q(\xi_2|\xi_1)a(\xi_1,\xi_2)h(\xi_2)d\xi_2 + (1 - r(\xi_1))h(\xi_1) \tag{4.7}$$

for every $\xi_1 \sim g$, where q is the proposal distribution, $a(\xi_k, \zeta_k) = \min\{\frac{q(\xi_k|\zeta_k)g(\zeta_k)}{q(\zeta_k|\xi_k)g(\xi_k)}, 1\}$; $r(\xi_k) = \int q(\zeta_k|\xi_k)a(\xi_k,\zeta_k)d\zeta_k$. Then,

$$\mathbb{E}_g[h(\xi_2)|\xi_1] = \mathbb{E}_g[h(\xi)]r(\xi_1) + (1 - r(\xi_1))h(\xi_1) \tag{4.8}$$

By definition, a harmonic function h has the property $\mathbb{E}_g[h(\xi_2)|\xi_1] = h(\xi_1)$. Therefore, equation (4.8) can be written as,

$$h(\xi_1) = \mathbb{E}_g[h(\xi)]r(\xi_1) + (1 - r(\xi_1))h(\xi_1) \tag{4.9}$$

[1] A *harmonic function* h for the chain $\xi_1, \ldots, \xi_N$ is defined as any measurable function that satisfies the following condition:

$$h(\xi_1) = \mathbb{E}[h(\xi_2)|\xi_1] = \mathbb{E}[h(\xi_N)|\xi_1] \tag{4.6}$$

$$\therefore \big\{\mathbb{E}_g[h(\xi)] - h(\xi_1)\big\} r(\xi_1) = 0 \tag{4.10}$$

Because $r(\xi_1) > 0$ (i.e.: the probability of moving to the next state is always positive),

$$h(\xi_1) = \mathbb{E}_g[h(\xi)] \tag{4.11}$$

Therefore, h is constant and the chain is Harris recurrent. □

In our proposed MCMC algorithm, the stationary distribution $g(\xi)$ is $|Q(\hat{x}, \xi)| f(\xi)$. As a result, the only difference between our method and the conventional MH method is the acceptance criterion, which now becomes $a(\xi_k, \zeta_k) = \min\{\frac{q(\xi_k|\zeta_k)|Q(\hat{x},\zeta_k)|f(\zeta_k)}{q(\zeta_k|\xi_k)|Q(\hat{x},\xi_k)|f(\xi_k)}, 1\}$. Following the same derivations above, we can conclude that the chain generated by our proposed MH method for SP algorithms is Harris recurrent.

So far, we have shown that our proposed MH method for SP algorithms satisfies the Detailed Balance Condition with the stationary distribution g^*; moreover, it is shown to be aperiodic and Harris recurrent. Therefore, according to Robert and Casella (2013), we obtain the following convergence results:

$$\int h(\xi) g(\xi) d\xi = \lim_{N\to\infty} \frac{1}{N} \sum_{i=1}^{N} h(\xi_i) \tag{4.12}$$

and

$$\lim_{N\to\infty} \| \int K^N(\xi, .)\mu(d\xi) - g^* \|_{TV} = 0 \tag{4.13}$$

for any initial distribution μ, and any initial state ξ, and K is the transition kernel of the proposed algorithm that has been defined in Equation (4.3) with the acceptance ratio $a(\xi_k, \zeta_k) = \min\{\frac{q(\xi_k|\zeta_k)|Q(\hat{x},\zeta_k)|f(\zeta_k)}{q(\zeta_k|\xi_k)|Q(\hat{x},\xi_k)|f(\xi_k)}, 1\}$. These results are important because they show that our proposed MH method for SP algorithms is an unbiased estimator.

In the next section, we will discuss about the convergence of the Kernel Density Estimation (KDE) for these generated samples. Based on the assumption of the Harris recurrence, Athreya and Atuncar (1998) have shown that the KDE for these samples converges and it is an unbiased estimator. Therefore, the samples drawn from this KDE distribution can be used to give a good approximation of the recourse function in SP problems.

4.2 Convergence of Kernel Density Estimation in the MCMC-IS algorithm

In the previous section, we have shown that our proposed MH method for SP algorithms generates an aperiodic, Harris recurrent chain with the stationary distribution that is optimal for the calculation of the recourse function in SP problems. The next step in our MCMC-IS algorithm is to use a KDE to construct the optimal IS distribution. There are two important reasons for doing so. Firstly, we can calculate the Radon-Nikodym derivative between the original distribution and the optimal IS distribution in order to correct the bias in the estimator. Secondly, we can generate more samples from this optimal IS distribution with a small overhead, which can enhance the accuracy and efficiency of our algorithm.

The convergence proof for KDE on a Harris recurrent chain has been shown in Athreya and Atuncar (1998). Therefore, we will only include the main result of that convergence proof for the completeness of the thesis. In the next section, we are going to introduce some additional background before we will discuss the main convergence proof in Section 4.2.2

4.2.1 Additional properties of Markov chain

In this section, we are going to state some important theorems and lemmas, which will be used in Section 4.2.2 to prove the convergence of KDE for a Harris-recurrent Markov chain.

An equivalent definition of *Harris recurrence* is given in Theorem 4.3. Based on this equivalent *Harris recurrence* definition, Athreya and Ney (1978) introduced the theory of *regeneration* as shown in Lemma 4.2. This is important because the Markov chain can be split into several small i.i.d. chains and hence the property of each chain can be investigated separately.

Theorem 4.3. ***(Equivalent Harris recurrence)*** *(Athreya and Ney (1978)) A Markov chain is called (A, ϵ, g, n_0) recurrent if there exists a set $A \in \Sigma$, a probability measure g on A, a real number $\epsilon > 0$, an integer $n_0 > 0$ such that:*

$$P_{\xi_0}(\tau_A < \infty) = P_{\xi_0}(\xi_n \in A \text{ for some } n \geq 1) = 1, \forall \xi_0 \in S \tag{4.14}$$

$$P_{\xi_0}(\xi_{n_0} \in E) = P^{n_0}(\xi_0, E) \geq \epsilon g(E), \forall E \subset A \tag{4.15}$$

where τ_A is the hitting time to the set A.

Lemma 4.2. ***(Regeneration)*** *(Athreya and Ney (1978)) If a Markov chain is $(A, \epsilon, g, 1)$ recurrent, there exists a random time T_Δ such that the evolution of process before and after T_Δ are independent. Each random tour is $\{\xi_j : j = T_\Delta^i, T_\Delta^i + 1, \ldots, T_\Delta^{i+1} - 1\}$, where T_Δ^i is the i^{th} hitting time to the state Δ. These tours are independent, identically distributed and all of them have distribution g.*

Based on the theory of *regeneration* (Athreya and Ney (1978)), the stationary measure and the calculation of a bounded measurable function for a Harris-recurrent Markov chain are given in Theorem 4.4 and Theorem **??** respectively.

Theorem 4.4. *(Athreya and Ney (1978)) Let $N_1 = T_\Delta^1$ be the regeneration time as defined in Lemma 4.2. For a Harris recurrent Markov chain ξ and a bounded measurable function h on S, and g is the stationary distribution,*

$$\int_S h d\nu = \mathbb{E}_g \sum_{i=0}^{N_1 - 1} h(\xi_i) \tag{4.16}$$

Moreover, for any Harris-recurrent Markov chain with the recurrence state Δ, the relationship between the mean and variance of the hitting time to state Δ, as well as the total number of visits to Δ and the total number of samples generated in the chain are given in Lemma 4.3.

Lemma 4.3. *(Asmussen (2003)) Let ξ be a Harris chain with a recurrence point Δ. Let $\lambda = \mathbb{E}\{t_\Delta\}$ and $\sigma^2 = Var(t_\Delta)$, where t_Δ is the average time to visit Δ. Let n be the number of visits to the state Δ when the chain moves from state 0 to state N. Then:*

$$\frac{n}{N} \to \frac{1}{\lambda} \text{ a.e. as } N \to \infty \tag{4.17}$$

$$\sqrt{N}(\frac{n}{N} - \frac{1}{\lambda}) \to N(0, \frac{\sigma^2}{\lambda^3}) \tag{4.18}$$

Finally, the convergence proof of KDE for a Harris-recurrent Markov chain requires a theorem about kernel function as given in Theorem 4.5.

Theorem 4.5. *(Rao (2014)) Assume that K satisfies the conditions of Equations 3.11 3.12 3.13. Define $g_N(\xi) = \int_{R^d} \frac{1}{h^d} K_H(\frac{z}{h}) g(\xi - z) dz$ where $h > 0$. When $h \to 0$ as $N \to \infty$,*

$$\lim_{N \to \infty} g_N(\xi) = g(\xi) \int_{R^d} K_H(z) dz \tag{4.19}$$

In this section, we have reviewed some important theorems and lemmas that prepare us to prove the convergence of KDE for a Harris recurrent Markov chain. As mentioned before, this proof has been shown in Athreya and Atuncar (1998) but we include it in the next section for the completeness of the thesis.

4.2.2 Convergence of Kernel Density Estimation for a Harris-recurrent Markov chain

In this section, we are going to discuss the convergence proof of KDE for a Harris-recurrent Markov chain. Given a Harris-recurrent Markov chain $\xi_1, \ldots, \xi_N$, where $\xi_1, \ldots, \xi_N \in R^d$, the KDE of this Chain is given as:

$$g_N(\xi) = \frac{1}{Nh^d} \sum_{i=0}^{N} K_H(\frac{\xi - \xi_i}{h}) \tag{4.20}$$

Based on Lemma 4.2, if a Markov chain is a Harris recurrent, we can find a number of random tours η_i (Athreya and Ney (1978) Athreya et al. (1996)), and each tour starts and ends at the state Δ. This Δ point can be considered as the breaking point such that the Markov chain before and after this point are independent. Hence, the properties of every small chain can be analyzed independently. We are going to prove the consistency of the KDE for a Harris recurrent chain as shown in Theorem 4.6.

Theorem 4.6. ***(Consistency)*** *Assume that K satisfies the conditions of Equations 3.11, 3.12, 3.13, if ξ is a Harris-recurrent Markov chain as given in Lemma 4.2 then:*

$$g_N(\xi) \to g(\xi) \text{ for almost all } \xi$$

as $h_N \to 0$ and $Nh_N \to \infty$; where $g_N(x)$ is defined in 4.20 and $g(x)$ is the true density. For the MCMC-IS algorithm, $g(x)$ is the optimal IS distribution of the recourse function.

Proof. (Athreya and Atuncar (1998)) Since ξ is a Harris recurrent Markov chain as defined in Lemma 4.2, we can find a sequence of random tours that are i.i.d. and return to a point Δ infinitely often (Athreya and Ney (1978) Athreya et al. (1996)). Define:

$$T_\Delta^i = \inf\{k > T_\Delta^{i-1} : \xi_k = \Delta\} \tag{4.21}$$

T_Δ^i is the index of sample that the Markov chain returns to the state Δ at time i. As a result, the chain from T_Δ^i to $T_\Delta^{i+1} - 1$ will form an i.i.d. Markov

chain with respect to the rest of the chain. Its properties therefore can be analyzed independently. The KDE for the chain from T_Δ^i to $T_\Delta^{i+1} - 1$ is given as:

$$\eta_i(\xi) = \sum_{j=T_\Delta^i}^{T_\Delta^{i+1}-1} \frac{1}{h^d} K_H(\frac{\xi - \xi_j}{h}) \tag{4.22}$$

According to Theorem 4.4 to Theorem 4.5, for all i and almost all ξ,

$$\begin{aligned}
\lim_{N\to\infty} \mathbb{E}_g\{\eta_i(\xi)\} &= \lim_{N\to\infty} \int_{R^d} \sum_{j=T_\Delta^i}^{T_\Delta^{i+1}-1} \frac{1}{h^d} K_H(\frac{\xi - u}{h}) g(u) du \\
&= \sum_{j=T_\Delta^i}^{T_\Delta^{i+1}-1} \lim_{N\to\infty} \int_{R^d} \frac{1}{h^d} K_H(\frac{\xi - u}{h}) g(u) du \\
&= \sum_{j=T_\Delta^i}^{T_\Delta^{i+1}-1} g(\xi) \int_{R^d} K_H(z) dz \\
&= \lambda g(\xi)
\end{aligned} \tag{4.23}$$

where $\lambda = \mathbb{E}\{t_\Delta\}$, which is the average time between T_Δ^i and $T_\Delta^{i+1} - 1$.

The KDE for a Harris-recurrent Markov chain (Equation 4.20) can be rewritten as:

$$g_N(\xi) = \frac{1}{Nh^d} \sum_{j=0}^{T_\Delta^1-1} K_H(\frac{\xi - \xi_j}{h}) + \frac{1}{Nh^d} \sum_{j=T_\Delta^1}^{T_\Delta^n-1} K_H(\frac{\xi - \xi_j}{h}) + \frac{1}{Nh^d} \sum_{j=T_\Delta^n}^{N} K_H(\frac{\xi - \xi_j}{h}) \tag{4.24}$$

where n is the last visit to the state Δ for the chain $\{\xi_0, \ldots, \xi_N\}$. As a result,

$$\begin{aligned}
\lim_{N\to\infty} \mathbb{E}_g\{g_N(\xi)\} &= \lim_{N\to\infty} \int_{R^d} \frac{1}{Nh^d} \sum_{j=0}^{T_\Delta^1-1} K_H(\frac{\xi-u}{h}) g(u)du + \\
&\quad \lim_{N\to\infty} \int_{R^d} \frac{1}{Nh^d} \sum_{j=T_\Delta^1}^{T_\Delta^n-1} K_H(\frac{\xi-u}{h}) g(u)du + \\
&\quad \lim_{N\to\infty} \int_{R^d} \frac{1}{Nh^d} \sum_{j=T_\Delta^n}^{N} K_H(\frac{\xi-u}{h}) g(u)du \\
&= \lim_{N\to\infty} \mathbb{E}\left\{ \frac{T_\Delta^1 g(\xi)}{N} + \frac{n\mathbb{E}\{\eta_i\}}{N} + \frac{(N-T_\Delta^n+1)g(\xi)}{N} \right\} \\
&= \lim_{N\to\infty} \{\frac{n\lambda g(\xi)}{N}\} \qquad (4.25) \\
&= g(\xi) \qquad (4.26)
\end{aligned}$$

The third equality can be derived due to the fact that the first and last term become zero as $N \to \infty$; and then according to Equation 4.23, we have Equation 4.25. Based on Lemma 4.3, we obtain Equation 4.26 □

In the next section, we are going to show the convergence of the SDDP algorithm when using the proposed MCMC-IS framework.

4.3 Convergence of Stochastic Dual Dynamic Programming with MCMC-IS algorithm

In Section 4.1, we have shown that our proposed MH method can generate an aperiodic, Harris recurrent chain with the stationary distribution that is optimal for the calculation of the recourse function in SP problems. Then, in Section 4.2, we have shown that the KDE for these samples converges and it is an unbiased estimator. Therefore, we can use it to calculate the accurate Radon-Nikodym derivative in order to correct the bias of IS estimator. In other words, we have shown that:

$$\mathbb{E}_g[Q(\hat{x},\xi)\frac{f(\xi)}{g(\xi)}] \to \mathbb{E}_f[Q(\hat{x},\xi)] \qquad (4.27)$$

where $\hat{x}$ is the previous stage decisions, f is the original distribution, g is the optimal IS distribution and ξ are random variables drawn the optimal IS distribution g.

This result shows that for a given $\hat{x}$, we can obtain the unbiased estimator of the recourse function. Now, our aim is to show that after a number of iterations, the SDDP algorithm with our proposed MCMC-IS framework will give us the optimal solutions x^*.

There is a large number of literature on the convergence of the SDDP algorithm (Philpott and Guan (2008) Shapiro (2011) Homem-de Mello et al. (2011) Homem-de Mello and Bayraksan (2014)). Based on the Assumption 4.1, the convergence of the SDDP algorithm with Sample Average Approximation (SAA) was shown in Theorem 4.7 (Shapiro (2011)).

Assumption 4.1. *(Shapiro (2011)) The master problem and subproblems at every stage have finite optimal values for all realizations of the data.*

Theorem 4.7. *(Shapiro (2011)) Suppose that in the forward steps of the SDDP algorithm the subsampling procedure is used and Assumption 4.1 holds. Then, in the backward steps, the basic optimal solutions are employed. After a sufficiently large number of backward and forward steps of the algorithm, we will obtain the optimal solutions with probability one.*

On the other hand, Homem-de Mello et al. (2011) combined the SDDP algorithm with two sampling strategies: Latin Hypercube Sampling and Randomized Quasi-Monte Carlo. They then proposed a different stopping criterion using hypothesis testing. Several other ways of assessing the quality of solutions for optimization algorithms were proposed in Homem-de Mello and Bayraksan (2014). In general, all of the stopping criteria involve the construction of the confidence interval to bound the optimality gap. When the bound on the optimality gap is small, we can conclude that the solution is of high quality.

Throughout this thesis, we will use the stopping criterion that was described in Shapiro (2011). This stopping criterion ensures that the optimization problem will be solved with the accuracy ϵ for the confidence of $(1-\alpha)$ (Shapiro (2011)). It works as follows:

Every iteration of the SDDP algorithm includes a backward and forward simulation. (Please refer back to Section 2.6) for the full explanation of the SDDP algorithm). During the backward simulation, we solve the subproblems at each stage in order to obtain the duals. The duals are then used to construct the supporting hyperplane (i.e.: cutting plane), which gives us the lower-bound linear approximation of the recourse function at each stage. Therefore, solving the first stage problem will give us the lower bound for the whole optimization problem. This is denoted as $\underline{\theta}$.

Then, in the forward simulation, we sample N paths from the first stage

to the final stage and calculate the "true" cost for each path:

$$\theta_i = \sum_{t=1}^{T} c_{ti}^T \hat{x}_{ti}, i = 1, \ldots, N \tag{4.28}$$

where c_{ti} and $\hat{x}_{ti}$ are the cost and trial decicion vectors at stage t for the path i. We then calculate the mean, $\bar{\theta}$, and the variance, $\hat{\sigma}_\theta^2$, of these paths:

$$\bar{\theta} = \frac{1}{N} \sum_{i=1}^{N} \theta_i \tag{4.29}$$

$$\hat{\sigma}_\theta^2 = \frac{1}{N-1} \sum_{i=1}^{N} (\theta_i - \bar{\theta})^2 \tag{4.30}$$

Then the confidence interval can be constructed as:

$$[\bar{\theta} - z_{\alpha/2}\hat{\sigma}_\theta/\sqrt{N}, \bar{\theta} + z_{\alpha/2}\hat{\sigma}_\theta/\sqrt{N}] \tag{4.31}$$

where z_α denotes the $(1-\alpha)$-quantile of the standard Normal distribution. For example: $z_{0.025} = 1.96$ shows the 95% confidence interval. If the difference between the upper confidence bound $\bar{\theta} + z_\alpha \hat{\sigma}_\theta \sqrt{N}$ and the lower bound $\underline{\theta}$ is less than a given accuracy level $\epsilon > 0$, the algorithm stops.

In summary, in this section, we have proved the convergence of our proposed MCMC-IS algorithm and explained the stopping criterion that will be used when our MCMC-IS algorithm combines with the SDDP algorithm to solve multistage optimization problems. In the next section, we will provide some practical guidelines for the implementation of Algorithm 4. After that, we will test the proposed algorithm on a newsvendor problem in Section 5.2. Then, we will test it on a large number of benchmark applications, which can be found in Ariyawansa and Felt (2004), in Section 5.3. Finally, we use our proposed algorithm to solve a large-scale optimization problem in the electric power industry, which integrates the planning of capacity expansion, unit commitment and maintenance scheduling into a unified model, in Chapter 6.

Chapter 5

Numerical experiments

5.1 Practical guidelines for MCMC-IS

In this chapter, we are going to provide some practical guidelines for the implementation of the Algorithm 4 (see Chapter 3). 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 5.2 and 5.3. It may be surprising that a small and constant number is sufficient even

for large-scale problems. In Section 5.2.2 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 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 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 approximate density using a simple Gaussian product kernel a leave-one-out cross-validation bandwidth estimator. The basic principle of leave-one-out cross-validation is that: Firstly, the density estimation is constructed from $N-1$ samples while leaving one sample out. This sample is assumed to be drawn from the same distribution as the other $N-1$ samples. It is, however, used as an out of sample in order to estimate the bandwidth such that we can have the maximum log-likelihood density estimation function. Each sample in the set of N samples will be left out in turn and for each turn, a KDE is constructed from the rest of $N-1$ samples. Eventually, the log-likelihood density estimation function is the average of N KDEs. The optimal bandwidth in this setting is chosen so that it gives the maximum log-likelihood density estimation function. For detailed explanation of the bandwidth selection, please refer to (Silverman (1998)) and (Jones et al. (1996)). 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 5.2.2.

5.2 Numerical results 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 as described in Section 2.7.1. In Sections 5.2.2 - 5.2.4, we illustrate differ-

ent sampling-related properties of MCMC-IS to provide insights into how MCMC-IS works and how it should be used in practice. In Section 5.2.5, 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 reasons why we chose to compare our proposed algorithm with CMC, QMC, and DGI are: CMC is one of most popular algorithms for solving SP problems (Birge and Louveaux (2011) Shapiro et al. (2009) Kleywegt et al. (2001)); QMC is the Variance Reduction method that has been applied and performed well in many problems (Homem-de Mello (2008) Drew (2007) Koivu (2005)); and DGI is one of the state-of-the-art Importance Sampling frameworks used to solve stochastic programs and it shares a similar philosophy with our proposed algorithm and therefore can be used as a good benchmark. The remaining numerical experiments in Section 5.2.7 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.

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

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. 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. Note that MCMC-IS uses a total of $M + M_r$ total function 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 5.1, 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 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.

5.2.2 The effect of the number of MCMC samples and the KDE bandwidth parameter

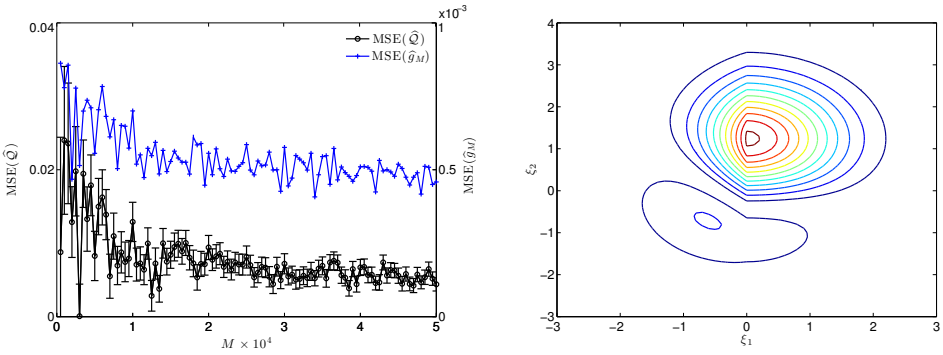
Our numerical experiments with the newsvendor problem suggest that a modest number of MCMC samples (M) can produce an approximate zero-variance distribution ($\widehat{g}_M$) that yields substantial variance reduction in our estimates of the recourse function.

As shown in Figure 5.1(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 5.1(b) and the contours of $\widehat{g}_M$ for different values of M in Figures 5.1(c)-5.1(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 5.1(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. These results were constructed with the first stage decision fixed at $\hat{x} = 50$.

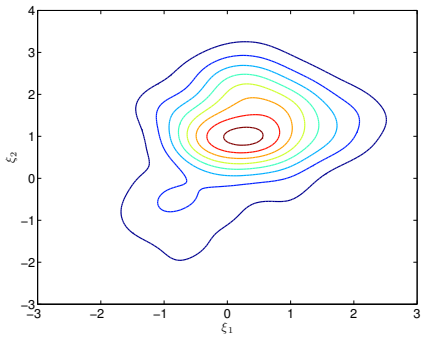
5.2.3 Adaptive sampling of the important regions

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

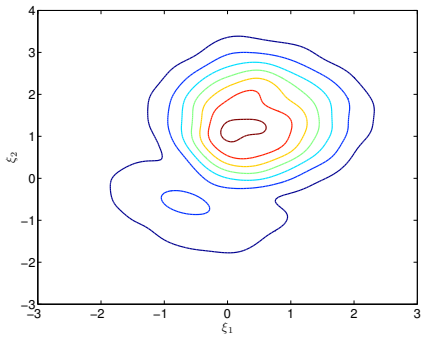


(a)

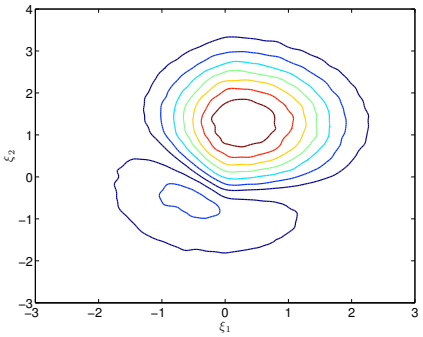
(b) g^*



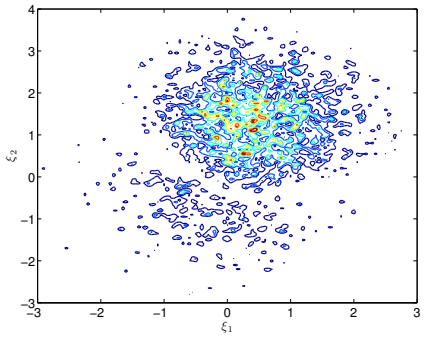
(c) $\widehat{g}_{500}$



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



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



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

Figure 5.1: (a) The majority of the gains in variance reduction and accuracy can be achieved for a small value of M. Note that the axis for MSE($\widehat{g}_M$) is on 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)

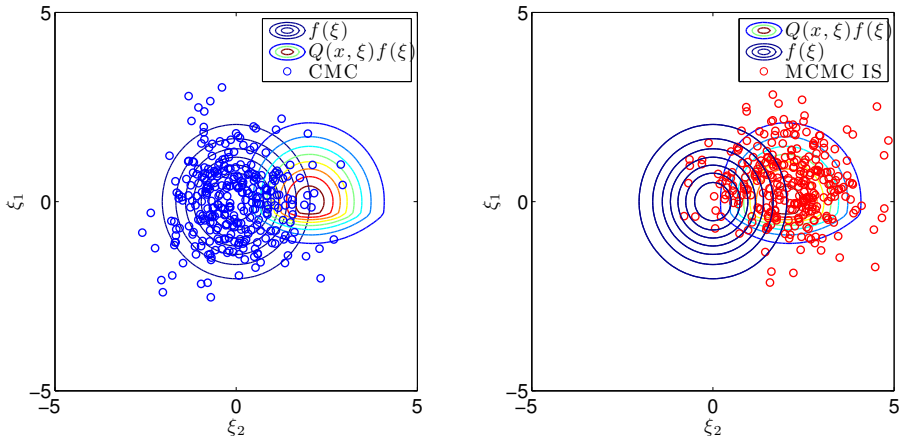


Figure 5.2: Comparison of points generated with the standard CMC approach, and MCMC-IS. Left: using MC sampling; Right: using importance sampling.

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 5.2 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 5.2 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 $\hat{x} = 50$.

5.2.4 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 5.3, 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.

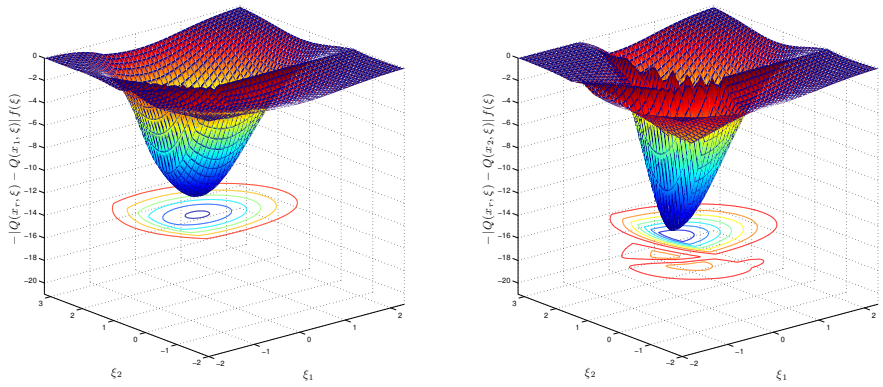


Figure 5.3: The absolute difference between an approximate zero-variance distribution constructed at $\widehat{x}_r = 50$ and two other approximate zero-variance distributions constructed at $\widehat{x}_1 = 10$ (left) and $\widehat{x}_2 = 100$ (right).

5.2.5 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 5.4(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 5.4(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 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 2.7.1. In this regime, MCMC-IS outperforms all other methods (Figure 5.4(c) and 5.4(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.

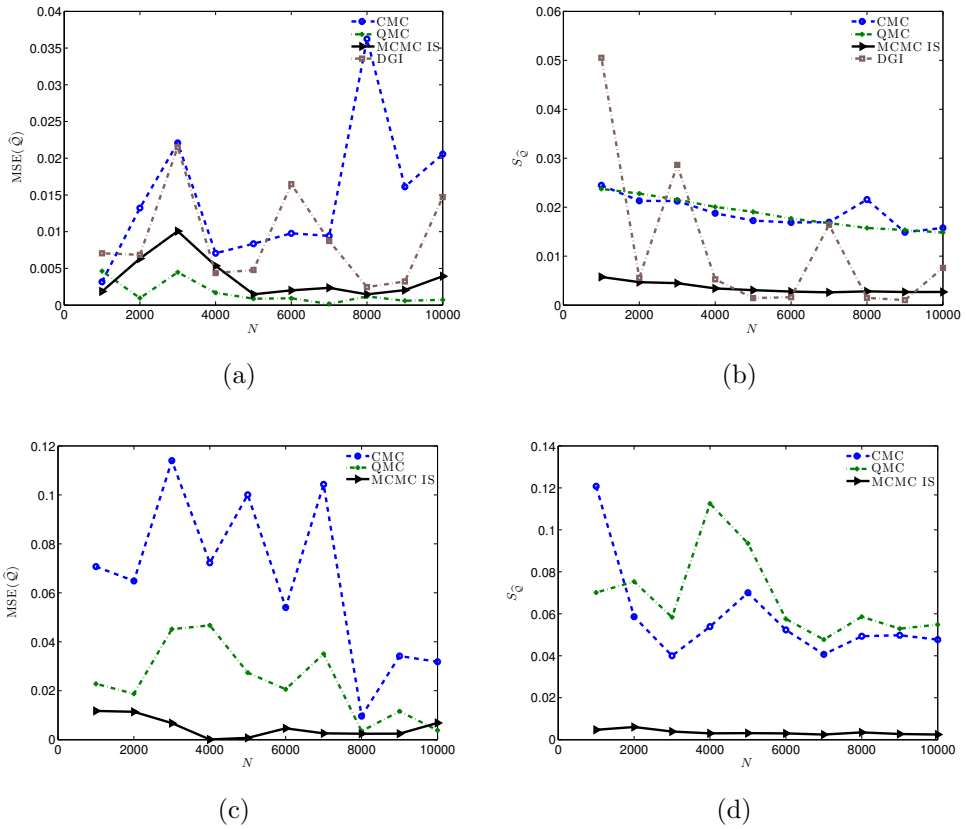


Figure 5.4: **Top:** Comparison of the accuracy and variance of estimates produced by different methods for a moderate-variance problem with $\sigma = 1$. **Bottom:** Comparison of the accuracy and variance of estimates produced by different methods for a higher-variance problem with $\sigma = 2$. Note that we omit the results for the IDG method when $\sigma = 2$ for clarity. The normalized values of $S_{\widehat{Q}}$ and MSE($\widehat{Q}$) for DGI are around 20% and 40% respectively.

5.2.6 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 (2.55) 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 5.5(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 5.2.4, 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 5.5(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 distribution $\widehat{g}_M$

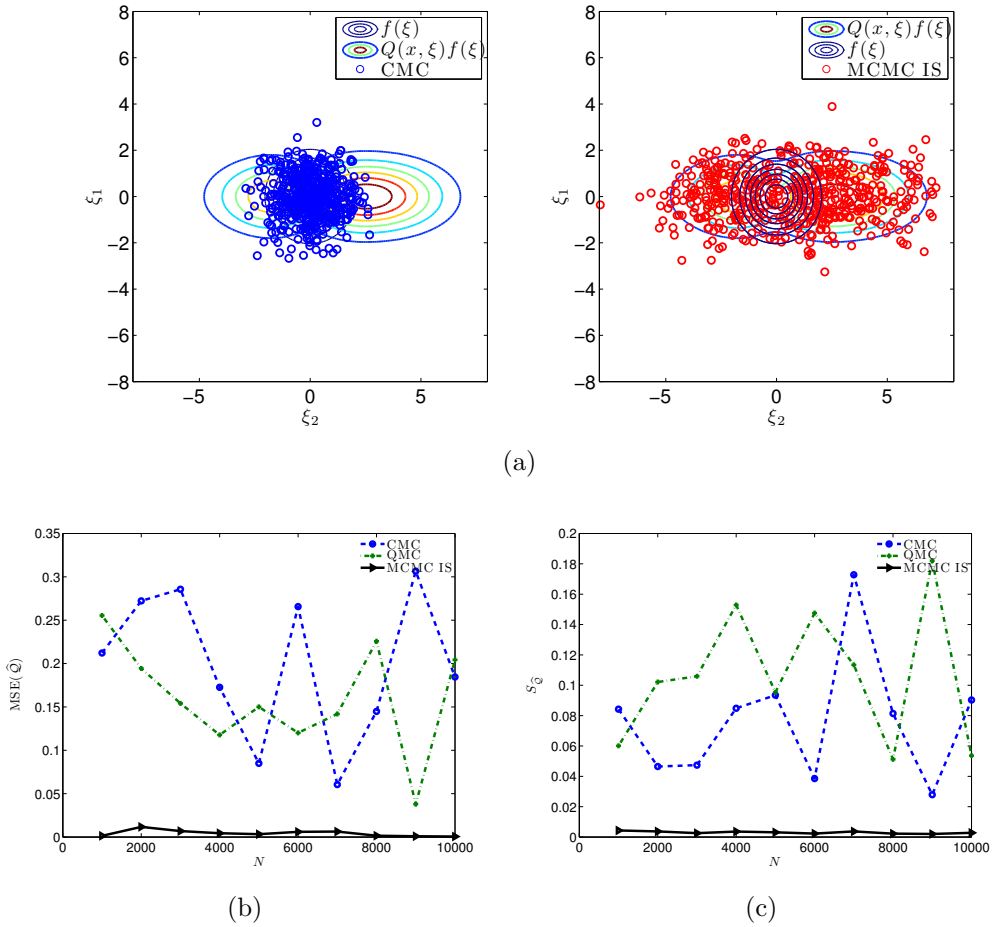


Figure 5.5: **Top**: Contours of a multimodal model. Samples generated using CMC are shown on the left and the samples from MCMC-IS are shown on the right. **Bottom:** Error and variance of estimates produced by different methods.

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 5.5(b)) and has less variance (Figure 5.5(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).

5.2.7 Accuracy and variance of MCMC-IS estimations 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 2.7.1, 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 2.7.1. 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 5.2.2 to 5.2.6, 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 5.6 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 5.6(a) - 5.6(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 2.7.1, 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 5.6(e) - 5.6(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 5.2.6, 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).

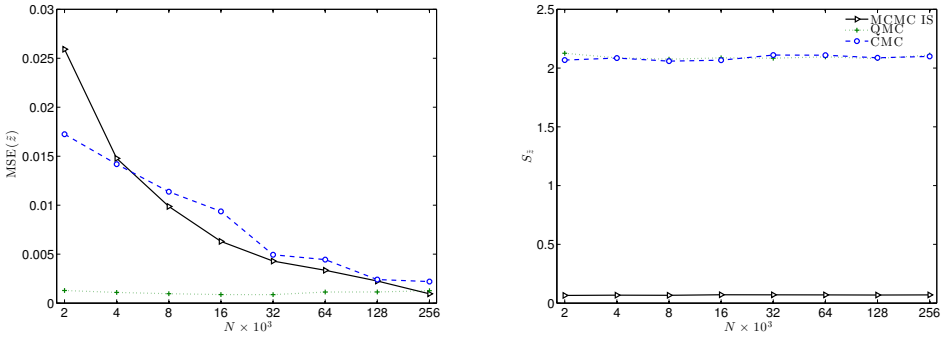
Our results confirm that the relative advantage of using MCMC-IS estimates depends on the inherent variance of the underlying stochastic programming model. In models where the uncertainty is modeled using a lower-variance distribution, MCMC-IS produces estimates that are just as accurate as the estimates 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.

5.3 Numerical results on a collection of test problems

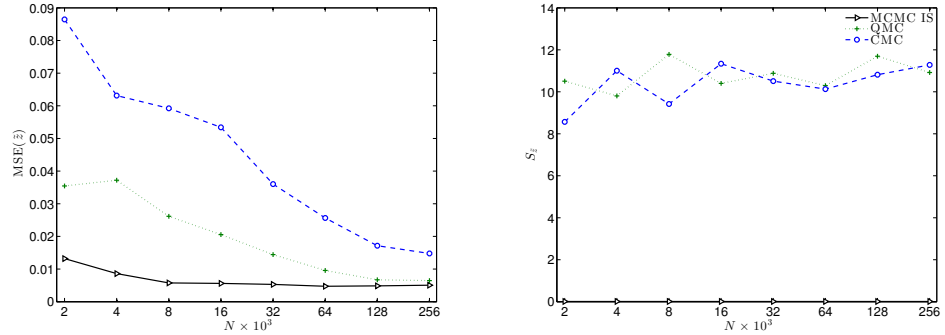
In this section, we demonstrate the performance of MCMC-IS when it is paired with a decomposition algorithm in order to solve a collection of benchmark stochastic programming models.

5.3.1 Overview of the test problems

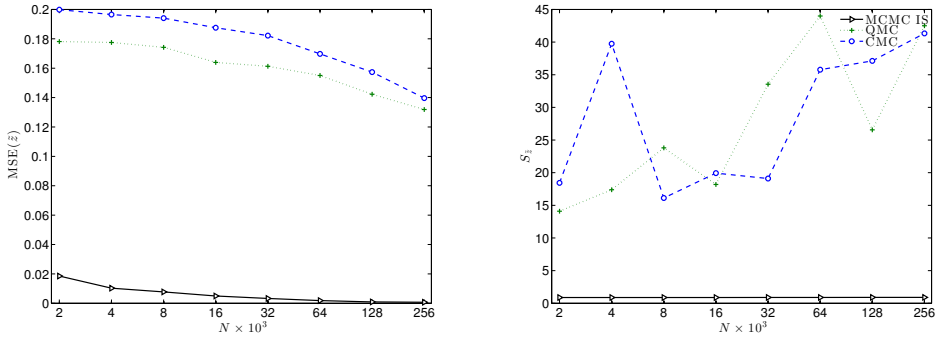
In order to verify that our findings from Section 5.2.7 generalize to stochastic programming models, we have based the numerical experiments in this section on a collection of 9 benchmark stochastic programming models from Ariyawansa and Felt (2004). We have specifically chosen these models due to the fact that they represent 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



(a) (b)



(c) (d)



(e) (f)

Figure 5.6: 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 $\sigma = 1$ (5.6(a) - 5.6(b)), a higher-variance lognormal distribution with $\sigma = 2$ (5.6(c) - 5.6(d)), and multimodal rare-event distribution (5.6(e) - 5.6(f))

addition, the models also pertain to decision-making problems across a wide range of application areas such as energy, finance, and telecommunications.

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 5.1: 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.

5.3.2 Details on the numerical experiments

As in Section 5.2.7, 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 function 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 8.4 in Appendix 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 which achieves a 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.

5.3.3 Accuracy and variance of the estimations

In Figure 5.7, 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 5.7(a)), higher-variance distributions (Figure 5.7(b)) and rare-event multimodal distributions (Figure 5.7(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 5.7(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 5.7(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 5.7(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 5.7(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.

5.4 When to use MCMC-IS in Stochastic Program

The proposed algorithm has an additional overhead when compared to CMC and QMC. The benefits of variance reduction are obvious from Figures 5.7(a)-5.7(c). In Figure 5.7(d) we showed that the algorithm is also efficient in the sense defined in the previous section. 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 5.2 below, 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 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 or not 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 in order 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 in order to accurately capture the uncertainties. Given the large error of other methods (see Figure 5.7(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.

# Samples	Median Overhead (%)(secs)			Median Var. Reduction (%)		
$(N \times 10^3)$	$\sigma = 1$	$\sigma = 2$	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%

Table 5.2: Computational overhead and variance reduction trade offs for MCMC-IS.

In the next section, we are going to use the proposed algorithm to solve

the capacity expansion planning in the electric power industry. This capacity expansion planning model is different to the existing models in such a way that it also includes the unit commitment problem and maintenance scheduling. The challenge is the problem becomes a very large scale optimization problem and it is very computationally intensive. We will propose different approaches for formulating the problems and then using the proposed algorithm to be able to solve the problem efficiently and accurately.

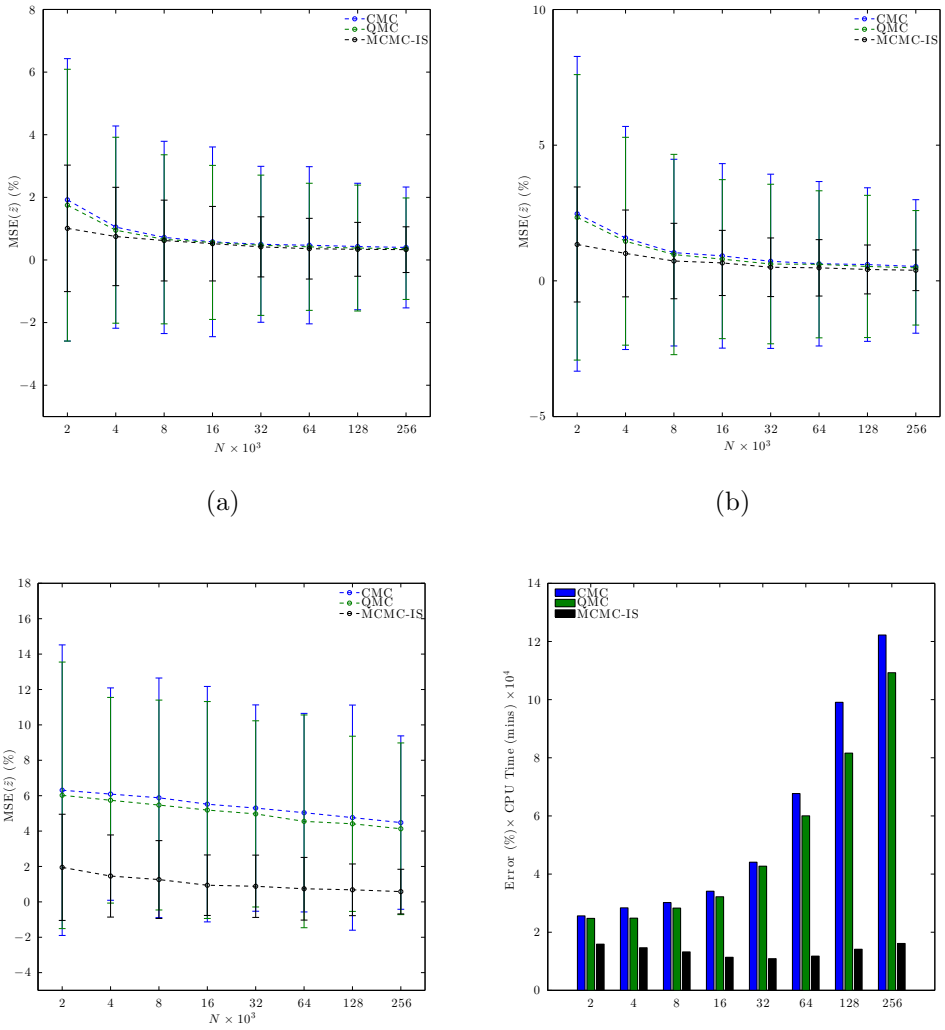


Figure 5.7: Median results with the Collection of Test Problems. (a) MSE($\tilde{z}$) for models with lower-variance distributions (b) MSE($\tilde{z}$) for models with higher-variance distributions (c) MSE($\tilde{z}$) for models with rare-event distribution. The error bars indicate the standard error associated with the solution obtained. (d) Error (%) × CPU Time (mins); for this plot we averaged the low variance, moderate variance and rare event results.

Chapter 6

Application: The capacity expansion planning in the electric power industry

In this chapter, we are going to present an optimization model that integrates the stochastic unit commitment problem, the maintenance scheduling and the capacity investment planning into a single framework as shown in Figure 6.1. This allows the investors to make the optimal decisions on the installed capacity of each generator to satisfy the electricity demand while miniziming the fixed cost, the operating cost and the maintenance cost for the power system over a long period of time. Some typical questions addressed in each model are shown in the corresponding box in Figure 6.1, and the overlapping box presents the interaction between models.

In particular, the aim of the unit commitment model is to ensure the system can produce sufficient electricity to meet the demand every hour. This requires a careful consideration of operational flexibility of the system and the systems operating cost in response to the uncertain demand. The demand can be predicted over the next few hours or they can be modelled as a statistical distribution based on historical data. The operational flexibility of the system depends on the flexibility of each generator, which is determined by many technical specifications such as the minimum and maximum capacity output, the ramping-up and ramping-down rate, the minimum operational time, etc. According to this information, the operators will coordinate the system and decide which generator should be turned on or off and how much power should be produced from them in order to achieve the overall minimal cost. The challenge arises further when the number of generators increases and a high penetration of renewable energy introduces more uncertainties into the system, requiring a significant flexibility. In addition, strict CO_2

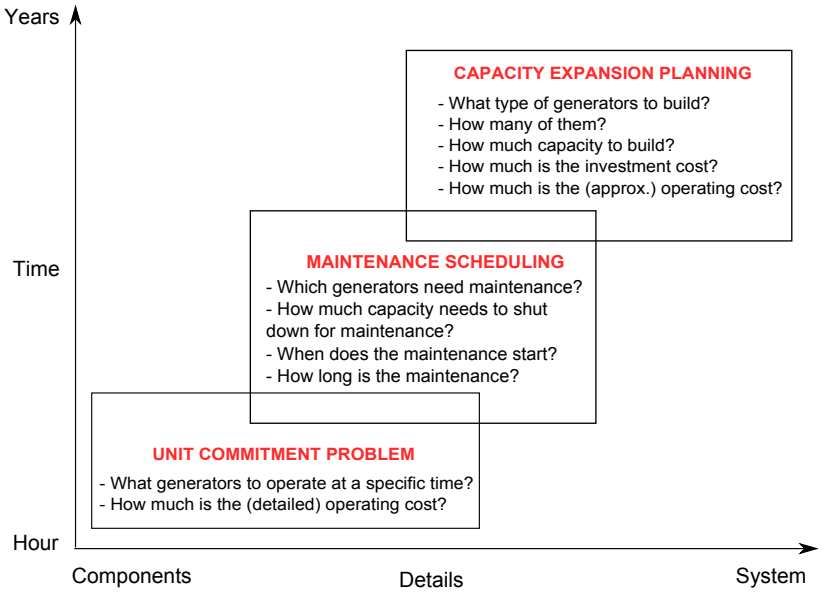


Figure 6.1: The unit commitment, maintenance scheduling and capacity expansion models are integrated into a single framework. Some typical questions addressed in each model are shown in the corresponding box. The overlapped box presents the interaction between models.

regulations encourage a large switch to the new low-CO_2 technologies to happen faster. The unit commitment problem has been studied over many years and it still remains one of the most active research areas in the operation research community due to its complexity and the benefits (i.e. cost saving) it can bring for every improvement in the quality of the model (Hobbs et al. (2001) Padhy (2004) Chao (2010) Palmintier (2013)). We will investigate this problem in detail in Section 6.1.1.

The next model that will be integrated into our unified framework is the maintenance scheduling model. The aim of this model is to minimize the maintenance cost while keeping the system reliable (Chattopadhyay et al. (1998)). The maintenance work for each generator is performed within a certain timescale and under the limited number of engineers, equipment, etc, to complete the task. We will investigate this problem in detail in Section 6.1.3.

The third model that will be integrated into our model is the capacity expansion planning problem. This model will provide the investors the optimal decisions on the capacity to build for every type of generator in order to satisfy the current and future demand while minimizing the capital and operating cost. The model usually includes a large number of different types

of generators and once being built, the generators will last for several years or decades. Therefore, this problem is usually a very large-scale multi-period optimization problem (Abilock et al. (1979) Botterud et al. (2005)). In order to reduce the complexity of the problem, many traditional models have simplified the unit-commitment problem by ignoring many technical constraints in the generators. This approach was acceptable in the traditional power system when the uncertainty such as demand changed slowly and predictably over time so the operational flexibility was not a big concern. Moreover, a strong correlation between the operational flexibility and the merit order plant dispatch made the system operators' task simple (Palmintier (2013)). For example: the traditional power system had coal generators run at all time, acting as the baseload power plants because they have the lowest operating cost. Only in the peak period, the demand was so high that some peakers (i.e. generators with high operational-flexibility such as Natural Gas fired combustion Gas Turbine (NGGT)) had to be switched on to quickly produce enough supply. These peakers usually had very high operating costs and hence it was cost-effective to use them only for a short period of time and in an emergency only (i.e. a few hours of peak demand). All of these assumptions are no longer satisfied in modern power systems networks. In modern power systems networks, many European countries and the United State (US) impose strict regulations on the greenhouse gas emissions and they also demand a high penetration level of renewable energy into the system (European Commission (2011) National Renewable Energy Laboratory (2012) Intergovernmental Panel on Climate Change (2014)). In particular, CO_2 is the main contributor to the greenhouse gas emissions so we will consider it as the only pollutant. The regulation imposes a heavy penalty cost for every ton of CO_2 emission. This gives incentives for electrical companies to switch their generation mix to low-CO_2 technologies such as coal with Carbon Capture and Sequestration (CCS) and gas generators. However, the operational flexibility of these generators are low as they have long minimum up/down time, small ramping-up, ramping-down rate, high minimum output level. Furthermore, they have high start-up cost. With the increasing use of renewable energy such as wind energy, the system now has to cope with more uncertainty, requiring it to be much more flexible such that the operational flexibility plays an important role in the modern power system can be demonstrated in the following scenario. During night time, the demand is usually low but wind output is high. If the output from renewable energy has to be at least 20% of the total generation, the output from another sources of energy such as coal and gas has to decrease significantly. As a result, some coal and gas generators will be fully switched off at night. On the next day, when the demand rises dramatically but the wind output plunges, there are

not enough coal and gas generators available to produce the supply. This leads to the risk of outage.

In fact, many recent reports such as the IEA (International Energy Agency), NERC (North American Electric Reliability Corporation) and research papers have emphasized the importance of capturing operational flexibility in the capacity expansion planning in the modern power system (International Energy Agency (2008) North American Electric Reliability Corporation (2009) Lannoye (2011)). However, due to the complexity of the combined model, most of the literature so far have solved them separately, which may lead to the sub-optimal solutions for the long-term capacity expansion problem (Palmintier (2013)).

The combination of three models was first proposed in Palmintier (2013). The paper showed that ignoring the operational flexibility may lead to an infeasible solution of unit commitment, which then increases the cost of any unserved-demand compensation. It can also result in the underestimation of CO_2 emission and thus having to pay for a high environmental penalty cost. In addition, in order to reduce the computational complexity, he proposed to cluster the generaters into groups. However, his model is a deterministic Mixed Integer Linear Programming (MILP) problem. In this paper, we will solve a multistage stochastic model, which also combines the unit commitment, maintenance scheduling and capacity investment planning together. The stochastic model will be more responsive to the uncertainties such as the electricity demand and the wind production, and hence it can provide a more robust solution. However, the problem can become intractable. To reduce the computational complexity, we propose the following methodological framework:

1. Formulate the problem as a Multistage Stochastic Linear Programming (MSLP) problem

2. Cluster similar generators together

3. Use an efficient optimization algorithm to solve the whole problem

The first step is to relax all of the integer variables in the problem so that we can exploit many well-established, powerful algorithms for solving multistage stochastic LP problems (Pereira and Pinto (1991) Birge and Louveaux (2011) Prekopa (1995) Higle and Sen (2013) Homem-de Mello and Bayraksan (2014)). The relaxation for the unit commitment problem is based on Weber (2005). According to Section 4.7 in Palmintier (2013), the relaxed problem (which is called as UcLp in Palmintier (2013)) gives a solution that is nearly identical to the originally MILP model in all but a few cases. The only difference is that the relaxed problem sometimes replaces the wind capacity with

another low-CO_2 technology such as the Natural Gas fired Combined Cycle Gas Turbine (NGCC) with a very small difference in the total cost. Apart from that, both models always produce enough power without any loss of load at any time and they can fully capture the generators operational flexibility such as their minimum up/down time, maximum ramping-up and ramping-down rate, the CO_2 emission of each generator, etc. In summary, "In terms of utility perspectives, the UcLp simplification does extremely well, producing generation mixes that cost essentially the same as those from the full Advanced model for all cases" - Page 165 (Palmintier (2013)). In addition, we are going to introduce some random variables in our formulation. The random variables include the demand and the wind output. They are modelled as lognormal distributions based on historical data. We will formulate the problem as the multistage stochastic program in Section 6.1.

In order to reduce the problem size, we propose to cluster generators with some common features together. The formalution for clustered problems is given in Section 6.2. (Palmintier (2013)) showed that using clustering in a deterministic unit-commitment problem can speed up the calculation up to 5000 times compared with the unclustered formulation while the error in the objective cost and solutions only increase by less than one percent. Compared with some other heuristic methods such as perturbation, merit order priority list, etc, in all cases, clustering is found to be the most effective approach Palmintier (2013).

In this paper, we will cluster the generators manually by two ways:

1. **By technology:** The generators are clustered according to the electricity generation technology. In particular, we consider four types of technology:

   - *Coal with Carbon Capture and Sequestration (Coal with CCS)*
   - *Natural Gas fired Combined Cycle Gas Turbine (NGCC)*
   - *Natural Gas fired Combustion Gas Turbine (NGGT)*
   - *Wind*

   The general features of each technology are described in Table 6.1.

2. **By technology and size:** The generators of each type of technology are then classified further into two groups: Large size and small size.

It is possible to use some clustering algorithms such as k-means, mean shift, Gaussian mixture model to automatically classify the generators (Celebi et al. (2013) Comaniciu and Meer (2002) Bishop (2006)). This will be investigated

	Coal with CCS	NGCC	NGGT	Wind
Capital cost	High	Medium	Lowest	Medium-High
Operating cost	Low	Medium	Highest	Very small
Operational flexibility	Low	Medium-High	Very high	Very low
Efficiency	Low	High	High	Medium

Table 6.1: Four types of technology and their main features

in our future work. Our focus in this section is to find the formulation and algorithm that are suitable for solving efficiently and accurately the capacity expansion planning problem while including an unit commitment and maintenance scheduling model.

A drawback of clustering is the possible increase in error of the solutions. This is because generators within a cluster are assumed to be homogeneous. In other words, they are assumed to have the same characteristics such as the minimum up time, the minimum down time, the maximum ramping-up rate, the maximum ramping-down rate, the minimum operational time, etc. In practice, any difference in the technical specifications of generators can lead to a difference in the system operation and therefore, it will affect the long-term investment decisions. We will investigate the tradeoff between the error of solutions and the computational time for different levels of aggregation in Section 6.3. Two methods that we will use to solve this problem are:

1. Stochastic Dual Dynamic Programming (SDDP) (Pereira and Pinto (1991))

2. MCMC-IS algorithm proposed in Section 3

The performance of different algorithms and formulations will be evaluated according to the percentage error of the total cost (i.e. the sum of capital cost and operations cost over a long period of time), the percentage error of the optimal first-stage solution and the computational time. Also, we measure the tradeoff between the computational time and the error. We will test every algorithm in two cases. The first case is when the random variables such as the demand and the wind output are drawn from a lognormal distribution with the standard deviation (SD) of one. The second case is when these random variables are drawn from a lognormal distribution with the SD of two. This situation is likely to happen in some places where the demand and wind output vary significantly. The details of the experiment and performance testing are given in Section 6.3. In the next section, we will introduce the problem formulation.

6.1 Formulation

We are going to formulate the unit-commitment problem in Section 6.1.1. Then, we propose an approach to scale effectively the time resolution from hour to season to year while keeping the structure of the problem unchanged and accessible for the integration of the maintenance scheduling and capacity expansion models. This scaling-time-resolution approach is given in Section 6.1.2. After that, we will introduce the maintenance scheduling formulation in Section 6.1.3, and then explain the capacity expansion formulation in Section 6.1.5.

6.1.1 Unit commitment problem

The ultimate aim of the unit-commitment problem is to find the optimal operations for the power system to supply the electricity at the lowest cost to meet the demand at all times. This can be achieved by obtaining the optimal generation mix among all of the generators in the system. Not only does the model has to determine what generator to switch on/off at a certain point of time, but also it is able to tell exactly how much power needs to be generated from every generator. As a result, it is crucial for the model to carefully take into consideration many technical constraints for every generator. Some of them may involve the dynamic link across several time periods. Moreover, the decisions at a later stage can be influenced by the previous decision variables. For example: a generator cannot be switched on and off immediately because of its minimum operation time. For the rest of this Section, we will explain in detail the constraints and the objective function in the unit-commitment problem.

The most important constraint in the unit-commitment problem is that the total generation must always exceed or equal the demand. Let $y_{g,t}$ be the generation of the generator g at time t; $D_t(\xi_1)$ is the uncertain demand at time t, which depends on the random variable $\xi_1 \sim N(\mu, \sigma)$. Then, the supply and demand constraint is given as follows:

$$\sum_{g} y_{g,t} \geq D_t(\xi_1) \tag{6.1}$$

In addition, the generation of the generator g cannot exceed its maximum capacity, K_g:

$$y_{g,t} \leq K_g \tag{6.2}$$

The commitment and start-up decisions for every generator are typically modelled as binary variables. However, the problem can easily become in-

tractable when the model includes a large number of generators. Many research papers have relaxed the problem by replacing the binary variables with the continuous ones. According to (Palmintier (2013)), the relaxed problem can provide identical results to the advanced binary-variables model while taking a significantly less amount of computational time. The relaxation can be done by defining an additional decision variable, known as the online capacity $y_{ONL,g,t}$ for every generator (Weber (2005)). This online capacity will give an upper bound for the generation, as shown in equation (6.3). The lower bound of the generation can be found by multiplying this online capacity $y_{ONL,g,t}$ with the minimum load factor, r_{MIN}, as shown in equation (6.4).

$$y_{g,t} \leq y_{ONL,g,t} \tag{6.3}$$

$$r_{MIN} * y_{ONL,g,t} \leq y_{g,t} \tag{6.4}$$

Moreover, the online capacity cannot exceed the maximum capacity that has been built:

$$y_{ONL,g,t} \leq K_g \tag{6.5}$$

For the renewable-energy generators, the online capacity is also limited by the amount of power available at that time. This amount of power, $P_{g,t}(\xi_2)$, is uncertain and can be described as a lognormal distribution with the random variable $\xi_2 \sim N(\mu, \sigma)$:

$$y_{ONL,g,t} \leq P_{g,t}(\xi_2), \forall g \in G_{renewable} \tag{6.6}$$

The start-up capacity of a generator is simply defined as the difference of the online capacity between time t and $t+1$:

$$y_{STU,g,t} \geq y_{ONL,g,t} - y_{ONL,g,t-1} \tag{6.7}$$

The cost for every start-up capacity unit is $c_{STU,g,t}$. Thus the total start-up cost for generator g at time t is given as:

$$C_{STU,g,t} = c_{STU,g,t} * y_{STU,g,t} \tag{6.8}$$

To calculate the operating cost, one common practice is to split it into two parts. The first part is to calculate the generation cost at the minimum output level. The second part is to calculate the marginal generation cost above the minimum output level (Weber (2005)). This is useful for modelling different levels of efficiency and cost when a generator performs at its full or partial capacity. Hence, the operating cost is calculated as:

$$C_{OP,g,t} = c_{f,g,t} * h_{0,g} * r_{MIN} * y_{ONL,g,t} + c_{f,g,t} * h_{m,g} * (y_{g,t} - r_{MIN} * y_{ONL,g,t})$$

$$= c_{f,g,t} * h_{m,g} * y_{g,t} + c_{f,g,t} * (h_{0,g} - h_{m,g}) * r_{MIN} * y_{ONL,g,t} \tag{6.9}$$

where $c_{f,g,t}$ is the fuel cost; $h_{0,g}$ is the heat rate at the minimum load factor; $h_{m,g}$ is the marginal heat rate between the minimum and full load. The maximum ramping-up rate, RU_g, is the maximum generation that can possibly increase from time t to $t+1$ (Li and Shahidehpour (2005)):

$$y_{g,t+1} - y_{g,t} \leq RU_g \tag{6.10}$$

The maximum ramping-down rate, RD_g, is the maximum generation that can possibly decrease from time t to $t+1$ (Li and Shahidehpour (2005)):

$$y_{g,t} - y_{g,t+1} \leq RD_g \tag{6.11}$$

The minimum operational time can be modelled by the difference between the online capacity between time t and $t+1$ such that this difference cannot exceed the online capacity during the last minimum operational time (Weber (2005)):

$$y_{ONL,g,t} - y_{ONL,g,t+1} \leq y_{ONL,g,\tau} \tag{6.12}$$

where $t - T_{OP,MIN,g} \leq \tau \leq t$ and $T_{OP,MIN,g}$ is the minimum operational time of generator g. The maximum start-up capacity is limited by the minimum offline capacity during the minimum shut-down time (Weber (2005)):

$$y_{STU,g,t} \leq K_g - y_{ONL,g,\tau} \tag{6.13}$$

where $\tau \geq t - T_{SD,MIN,g}$ and $T_{SD,MIN,g}$ is the minimum shut-down time of generator g.
The objective of the unit-commitment problem is to find the minimum generation cost:

$$\min \sum_t \sum_g C_{STU,g,t} + C_{OP,g,t} \tag{6.14}$$

In the next section, we will propose an approach to scale up the time effectively from hour to season to year, so that the maintenance scheduling and capacity expansion problems can be integrated into this model.

6.1.2 Scaling time

We make the following assumption: the demand for electricity throughout a year can be approximated by the demand of four seasons. During a season, the daily demand is assumed to have the same pattern for twenty four hours. Therefore, instead of using $8760 (= 365 * 24)$ time periods for modelling a-year

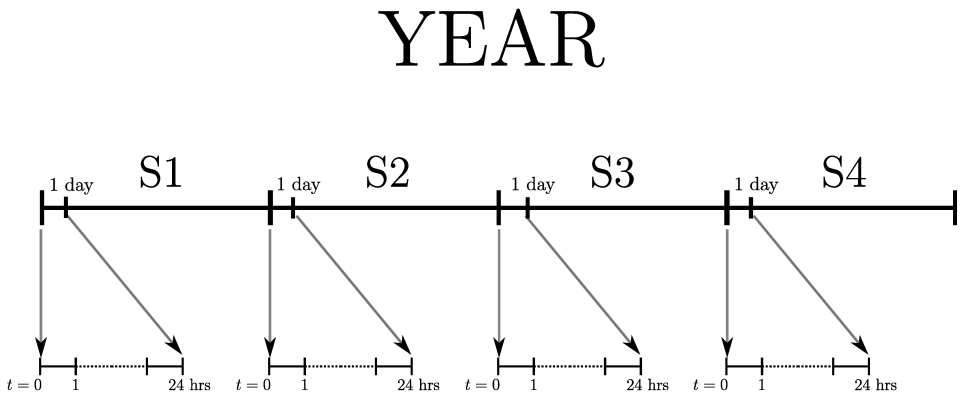


Figure 6.2: Multiscale explanation

operation of the power system, the problem is now reduced to $96(= 4 * 24)$ time periods. The scaling approach is demonstrated in Figure 6.2. This assumption was rigorously justified in Parpas and Webster (2013).

In this case, all of the formulas in the Section 6.1.1 are still applied. The only change to be made is that: the variables with subscript t now have an additional subscript s. For example: $y_{g,t}$ now becomes $y_{g,s,t}$ in order to denote the generation of generator g in season s at hour t.

In the next section, we will introduce the maintenance scheduling formulation. In this problem, the decisions are made for every season and they will affect the decisions in the unit-commitment problem.

6.1.3 Maintenance scheduling problem

Maintenance scheduling is an important process in the operation of the power system. Every generator must have a regular maintenance service to perform efficiently and safely for a long time. During the maintenance work, the capacity of the on-maintenance generators is reduced so it is important to balance how much capacity and when the maintenance work for each generator should be carried out without affecting the system's reliability and the maintenance cost. The maintenance capacity is also subject to the availability of engineers, equipment, etc. One common practice is that: The maintenance work for a generator will be performed continuously until its full capacity. This allows the electrical company to pay only once for engineers, equipment rental and management fee, etc (Chattopadhyay et al. (1995) Chattopadhyay et al. (1998)). Let $y_{maint,g,s}$ be the maintenance capacity of generator g in season s. The generation, $y_{g,s,t}$, is now given as:

$$0 \leq y_{g,s,t} \leq K_g - y_{maint,g,s} \tag{6.15}$$

The limited number of engineers, equipment, etc, is described in the coefficient $a_{limited-resource,g}$:

$$y_{maint,g,s} \leq a_{limited-resource,g} * K_g \tag{6.16}$$

For example: Due to the small number of engineers, the maintenance capacity can only perform up to 50% of the capacity of generator g in season s. In this case, $a_{limited-resource,g} = 50\%$. If the maintenance work for a generator cannot finish in a season, it will be likely to continue in the next season for the reason explained above. In that case, the constraint is given as:

$$\sum_{\tau=s-S_{maint,g}}^{s} y_{maint,g,\tau} = K_g \tag{6.17}$$

Where $S_{maint,g}$ is the number of seasons required for maintaining generator g.
The maintenance cost of generator g in season s is given as:

$$C_{maint,g,s} = c_{maint,g,s} * y_{maint,g,s} \tag{6.18}$$

Where $c_{maint,g,s}$ is the maintenance cost for every maintenance capacity unit. The objective of the maintenance scheduling is to find the minimum maintenance cost:

$$\min \sum_{g} \sum_{s} C_{maint,g,s} \tag{6.19}$$

In the next section, we will take into account the effects of CO_2 emission, the integration of renewable energy and the charge for non-served energy.

6.1.4 Carbon Dioxide emission, renewable energy integration, non-served energy penalty

In the previous sections, we introduced the unit-commitment and maintenance scheduling formulation. In this section, we will take into account the cost of CO_2 emission, the cost of unmet renewable energy and the cost of non-served energy over a long period of time.

The cost of CO_2 emission from generator g in season s at hour t is given as:

$$C_{CO_2,s,t} = \sum_{g} l_s * c_{CO_2} * a_{g,CO_2} * y_{g,s,t} \tag{6.20}$$

where l_s is the number of days in season s, c_{CO_2} is the cost for every ton of CO_2 emission, a_{g,CO_2} is the CO_2 emission from generator g to produce one

power unit.
The Renewable Portfolio Standard (RPS) requires the power system to have at least a certain level of renewable-energy integration. For example: the generation from renewable-energy sources must share at least 20% of the total generation at any point of time (Wiser and Barbose (2008)). The constraint for RPS requirement is given as:

$$E_{RPS-unmet,s,t} + \sum_{g \in G_{renewable}} y_{g,s,t} * l_s * d_t \geq a_{RPS} * \sum_{g \in G} y_{g,s,t} * l_s * d_t \quad (6.21)$$

where d_t is the duration of time t; a_{RPS} is the required renewable-energy penetration level, expressed in percentage; $E_{RPS-unmet,s,t}$ is the unmet renewable energy. In Equation 6.21, the power generation $y_{g,s,t}$ is multiplied with the time d_t to obtain the energy. In this paper, $d_t = 1$. There is a cost $c_{rps-unmet}$ for every unmet renewable-energy unit. Hence, the cost for the unmet renewable energy is given as:

$$C_{RPS-unmet,s,t} = c_{rps-unmet} * E_{RPS-unmet,s,t} \quad (6.22)$$

In some rare events (for example: The demand surges and the supply cannot meet the demand after trying every possible solution), the electrical company has to pay a penalty cost for the unserved energy, $E_{unserved,s,t}$:

$$C_{unserved,s,t} = c_{unserved} * E_{unserved,s,t} \quad (6.23)$$

where $c_{unserved}$ is the cost of every unserved energy unit; $E_{unserved,s,t}$ is given in the following equation:

$$E_{unserved,s,t} + \sum_{g} y_{g,s,t} * l_s * d_t = D_{s,t} * l_s * d_t \quad (6.24)$$

In the next section, we will introduce the capacity expansion formulation. It will be seen that there are strong interactions between the decisions in the unit-commitment problem, the maintenance scheduling problem and the capacity expansion problem.

6.1.5 Capacity expansion problem

The objective of the capacity expansion problem is to provide investors the optimal decisions on what types of generator and how much capacity to build to maximize their profit. In other words, the aim is to minimize the capital cost and the expected operating cost over the next several years. The capital cost for building a generator g is given by:

$$c_g * n_g \quad (6.25)$$

Where c_g is the capital cost of building a generator g; n_g is number of generator g. The constraint for n_g is:

$$0 \leq n_g \leq 1 \tag{6.26}$$

In this case, $n_g = 1$ suggests that the full capacity of generator g should be built; $n_g = 0$ shows that generator g should not be built; and if the value is somewhere between zero and one, it implies that a proportion of the capacity of generator g should be constructed. As a result, the capacity constraint for a generator (Equation (6.5)) is modified as follows:

$$y_{ONL,g,t} \leq n_g K_g \tag{6.27}$$

The operating cost is the expected sum of the start-up cost, the cost of electricity generation, the maintenance cost, the cost of RSP unmet energy and the cost of unserved demand over the next several years:

$$C_{total} = \min \sum_{g} (n_g * c_g) + \mathbb{E}_{\xi} \Bigg\{ \sum_{s} \sum_{g} C_{maint,g,s} + \sum_{s} \sum_{t} \sum_{g} \{ l_s * (C_{STU,g,s,t} + C_{OP,g,s,t}) + C_{CO_2,s,t} + C_{RPS-unmet,s,t} + C_{unserved,s,t} \} \Bigg\} \tag{6.28}$$

As we have seen so far, the formulation combining three models into a single framework will have a large number of decision variables and constraints. In the next section, we will explain the clustering method to reduce the problem size.

6.2 Clustered formulation

In this section, we are going to reduce the number of decision variables and constraints in our optimization problem using the clustering technique. The generators of similar characteristics (e.g.: technology, size, age, etc) are grouped together to be considered as one type of generator only. Therefore, the dimension of the problem shrinks down proportionally to the aggregration (clustering) level. We note that this dimensionality reduction is applied to the problem at every time period. The benefit of clustering, hence, will be compounded over the number of time periods, making the problem much more tractable. The drawback is that the error in the objective cost and optimal solutions may rise. The extent of the increase depends on the similarity of the generators in a cluster, which depends on the clustering features that the system operators or experts decide to use.

The clustered formulation requires only a small number of changes to the formulation in Section 6.1. In particular, the changes occur in the equations involving the capacity, the available renewable-energy capacity, the maximum ramping-up and ramping-down rate. In the unclustered formulation, these quantities represent only one generator; whereas in the clustered formulation, they are used for describing a group of generators. As a result, the capacity in the clustered formulation, $K_{\hat{g}}$, is equal to the sum of all of the generators capacity in the clustered set $\hat{G}$:

$$K_{\hat{g}} = \sum_{g \in \hat{G}} K_g = \sum_{g \in \hat{G}} n_g K_g \tag{6.29}$$

where K_g is the capacity of one generator as defined in Equation 6.2. In this paper, we use the "ˆ" notation to identify a clustered variable.
Similarly, the available renewable-energy capacity, the maximum ramping-up and ramping-down rate in the clustered formulation are given as:

$$P_{\hat{g},s,t}(\xi_2) = \sum_{g \in \hat{G}} P_{g,s,t}(\xi_2) \tag{6.30}$$

$$RU_{\hat{g}} = \sum_{g \in \hat{G}} RU_g \tag{6.31}$$

$$RD_{\hat{g}} = \sum_{g \in \hat{G}} RD_g \tag{6.32}$$

where $P_{g,t}(\xi_2)$, RU_g, RD_g are the available renewable-energy capacity, the maximum ramping-up rate, the maximum ramping-down rate for one generator as defined in Equation 6.6, 6.10 and 6.11 respectively.
In addition, the constraint for the number of generators is given by:

$$0 \leq n_{\hat{g}} \leq N_{MAX,\hat{g}} \tag{6.33}$$

where $N_{MAX,\hat{g}}$ is the maximum number of generators in the cluster. For example: If five generators are clustered together into a single generator, $N_{MAX,\hat{g}} = 5$.
On the other hand, the parameters such as the minimum operational time $T_{OP,MIN,g}$, the minimum shut-down time $T_{SD,MIN,g}$; the heat rate at the minimum load $h_{0,g}$, the marginal heat rate $h_{m,g}$, the minimum load factor r_{MIN}, the start-up cost per unit $c_{STU,g,t}$, the generation cost per unit $c_{f,g,t}$, the maintenance cost per unit $c_{maint,g,t}$, the capital cost for building a generator c_g, are the mean value of the generators in the cluster.

Other parameters such as the cost for every ton of CO_2 emission, c_{CO_2}, the cost of every unserved-demand unit, $c_{unserved}$, the cost of every unmet RPS-energy unit, $c_{RPS-unmet}$, the duration at time t, dt, the number of days in a seasons, l_s, which are set either by the government or nature facts, will be kept the same in the clustered formulation.
As a result, the clustered formulation is given as:

$$C_{total} = \min \sum_{\hat{g}} (n_{\hat{g}} * c_{\hat{g}}) + \mathbb{E}_{\xi} \Bigg\{ \sum_{s} \sum_{\hat{g}} \hat{C}_{maint,\hat{g},s} + \sum_{s} \sum_{t} \sum_{\hat{g}} \{ l_s * (\hat{C}_{STU,\hat{g},s,t} + \hat{C}_{OP,\hat{g},s,t}) + \hat{C}_{CO_2,s,t} + \hat{C}_{RPS-unmet,s,t} + \hat{C}_{unserved,s,t} \} \Bigg\} \quad (6.34)$$

Subject to:

$$\sum_{\hat{g}} y_{\hat{g},s,t} \geq D_{s,t}(\xi_1) \quad (6.35)$$

$$y_{\hat{g},s,t} \leq K_{\hat{g}} \quad (6.36)$$

$$y_{\hat{g},s,t} \leq y_{ONL,\hat{g},s,t} \quad (6.37)$$

$$y_{ONL,\hat{g},s,t} \leq P_{\hat{g},s,t}(\xi_2), \forall \hat{g} \in \hat{G}_{renewable} \quad (6.38)$$

$$r_{MIN} * y_{ONL,\hat{g},s,t} \leq y_{\hat{g},s,t} \quad (6.39)$$

$$y_{ONL,\hat{g},s,t} \leq K_{\hat{g}} \quad (6.40)$$

$$y_{STU,\hat{g},s,t} \geq y_{ONL,\hat{g},s,t} - y_{ONL,\hat{g},s,t-1} \quad (6.41)$$

$$\hat{C}_{STU,\hat{g},s,t} = c_{STU,\hat{g},s,t} * y_{STU,\hat{g},s,t} \quad (6.42)$$

$$\hat{C}_{OP,\hat{g},s,t} = c_{f,\hat{g},s,t} * h_{0,\hat{g}} * r_{MIN} * y_{ONL,\hat{g},s,t} + c_{f,\hat{g},s,t} * h_{m,\hat{g}} * (y_{\hat{g},s,t} - r_{MIN} * y_{ONL,\hat{g},s,t})$$

$$= c_{f,\hat{g},s,t} * h_{m,\hat{g}} * y_{\hat{g},s,t} + c_{f,\hat{g},s,t} * (h_{0,\hat{g}} - h_{m,\hat{g}}) * r_{MIN} * y_{ONL,\hat{g},s,t} \quad (6.43)$$

$$y_{ONL,\hat{g},s,t} - y_{ONL,\hat{g},s,t+1} \leq y_{ONL,\hat{g},\tau} \tag{6.44}$$

Where $\tau \geq t - T_{OP,MIN,\hat{g}}$

$$y_{STU,\hat{g},s,t} \leq K_{\hat{g}} - y_{ONL,\hat{g},\tau} \tag{6.45}$$

Where $\tau \geq t - T_{SD,MIN,\hat{g}}$

$$y_{\hat{g},s,t+1} - y_{\hat{g},s,t} \leq RU_{\hat{g}} \tag{6.46}$$

$$y_{\hat{g},s,t} - y_{\hat{g},s,t+1} \leq RD_{\hat{g}} \tag{6.47}$$

$$0 \leq y_{\hat{g},s,t} \leq K_{\hat{g}} - y_{maint,\hat{g},s} \tag{6.48}$$

$$y_{maint,\hat{g},s} \leq a_{limited-resource,\hat{g}} * K_{\hat{g}} \tag{6.49}$$

$$\sum_{\tau=s-S_{maint,\hat{g}}}^{s} y_{maint,\hat{g},\tau} = K_{\hat{g}} \tag{6.50}$$

$$\hat{C}_{maint,\hat{g},s} = c_{maint,\hat{g},s} * y_{maint,\hat{g},s} \tag{6.51}$$

$$\hat{C}_{CO_2,s,t} = \sum_{\hat{g}} l_s * c_{CO_2} * a_{\hat{g},CO_2} * y_{\hat{g},s,t} \tag{6.52}$$

$$\hat{E}_{RPS-unmet,s,t} + \sum_{\hat{g} \in \hat{G}_{renewable}} y_{g,s,t} * l_s * d_t \geq a_{RPS} * \sum_{g \in G} y_{g,s,t} * l_s * d_t \tag{6.53}$$

$$\hat{C}_{RPS-unmet,s,t} = c_{rps-unmet} * \hat{E}_{RPS-unmet,s,t} \tag{6.54}$$

$$\hat{E}_{unserved,s,t} + \sum_{\hat{g}} y_{\hat{g},s,t} * l_s * d_t = D_{s,t} * l_s * d_t \tag{6.55}$$

$$\hat{C}_{unserved,s,t} = c_{unserved} * \hat{E}_{unserved,s,t} \tag{6.56}$$

$$0 \leq n_{\hat{g}} \leq N_{MAX,\hat{g}} \tag{6.57}$$

In the next section, we are going to explain the numerical results of different formulations proposed in Section 6.1 and Section 6.2 with two different methods (SDDP and MCMC-IS). The pseudo code for the SDDP and MCMC-IS can be found in Algorithm 5.

Algorithm 5 SDDP and MCMC-IS algorithm

BACKWARD SIMULATION
for $t = T$ to 1 **do**
SDDP: Generate random variables using MC
OR
MCMC-IS: Generate random variables using MCMC. Then construct the Importance Sampling distribution by KDE based on the generated samples
Solve the subproblems with the generated samples $\rightarrow$ Obtain the duals
Compute Benders cuts
Add the cuts to the previous stage
end for
Set LB = The objective value at $t = 1$
FORWARD SIMULATION
Sample M forward paths
for $m = 1$ to M **do**
for $t = 1$ to T **do**
Solve the $f(t) + \theta(t+1)$, where $\theta(t+1)$ is the approximate cost-to-go function value
Store the solution as the trial solution for the next-iteration backward simulation
end for
Set the upper bound of path m = The objective cost
end for
Calculate the mean and standard deviation of M paths. Denote them as μ_M and σ_M.
STOPPING CRITERION
if $\mu_M - \frac{1.96*\sigma_M}{\sqrt{M}} \leq LB \leq \mu_M + \frac{1.96*\sigma_M}{\sqrt{M}}$ **then**
STOP
else
Return to **BACKWARD SIMULATION**
end if

6.3 Numerical results

In this section, we are going to compare the performance of different formulations and methods. The comparisons are based on three criteria:

1. The CPU time
2. The % error
3. The trade-off between the CPU time and the % error

After that, we are going to analyze the optimal decisions obtained by different methods.

In the previous sections, we reduced the complexity of the problem by relaxing the binary decision variables. Then we formulated the whole problem as a multistage stochastic programming problem in Section 6.1 so that we can exploit its special structure and solve it efficiently by a decomposition algorithm. After that, we proposed the clustered formulation to further reduce the number of decision variables and constraints in Section 6.2.

As mentioned before, we use the resolution of hours for modelling the unit-commitment problem because any rapid change in the hourly demand and wind output can make a heavy impact on the unit commitment and total cost, and therefore they can change significantly the long-term investment decision. We choose the resolution of years for modelling the investment planning because the investors may want to look longer towards the future to make their decisions. The resolution for the maintenance scheduling is seasonal because the maintenance decisions are usually made quarterly.

In our experiments, we will test different methods on a number of different time periods: 48, 96, 144 and 192, which are corresponding to 2, 4, 6 and 8 seasons respectively.

In the experiment, we will test six methods (which are the combination of different types of clustering and algorithm):

1. Unclustered with SDDP
2. Unclustered with MCMC-IS
3. Clustered-by-Tech with SDDP
4. Clustered-by-tech with MCMC-IS
5. Clustered-by-Tech-and-Size with SDDP
6. Clustered-by-Tech-and-Size with MCMC-IS

For ease of exposition, the method $1, 3$ and 5 will be written as:

1= Unclustered

3= Clustered-by-Tech

5= Clustered-by-Tech-and-Size

Three metrics will be used to compare the performance between different methods:

1. The CPU time
2. The % error, which is given as:

$$\% \text{ error} = \left| \frac{C_{\text{total}} - C^*_{\text{total}}}{C^*_{\text{total}}} \right| \times 100 \tag{6.58}$$

   where C^*_{total} is the *true* objective cost, which is obtained by performing the SDDP algorithm on the unclustered formulation for a very large number of samples. In this case, we used 1000 samples for every time period. C_{total} is the objective cost obtained by the tested algorithm using 100 samples. In the SDDP algorithm, the samples were generated by Crude Monte Carlo method. In the MCMC-IS, the samples were generated by the MCMC algorithm. Due to the acceptance-rejection step in the MCMC, there were more than 100 samples generated during the process. However, we ensure that the number of function evaluations are always kept the same for all of the methods in order to have a fair comparison.

3. The "Relative Efficiency", which is defined as:

$$\text{Relative Efficiency} = \text{CPU running time} \times \% \text{ error} \tag{6.59}$$

This "Relative Efficiency" metric measures the tradeoff between the CPU time and the percent error. The small value indicates that the algorithm takes a small amount of time for a given percent error.
The algorithms were implemented in MATLAB, in which many heavily computational functions, including the SDDP, MCMC, KDE were implemented in MEX/C++ in order to improve the efficiency. The MCMC algorithm used in this paper was proposed in Haario et al. (2001). The KDE algorithm was taken from this source: `http://www.ics.uci.edu/~ihler/code/kde.html`
The generators specifications and fuel cost were obtained from U.S. Energy Information Administration (2010) Northwest Power and Conservation

Council (2010) U.S. Environmental Protection Agency (2010) Palmintier (2013). The random variables in our model are: the demand and the wind output used in Equation 6.1, 6.6, 6.30, 6.35, 6.38. We assumed that they had the lognormal distributions:

$$D_t(\xi_1) = a * \exp(b * \xi_1) \tag{6.60}$$

$$P_{g,t}(\xi_2) = c * \exp(d * \xi_2) \tag{6.61}$$

Where $\xi_1, \xi_2 \sim N(\mu, \sigma^2)$; a and b are chosen to closely match the load data of the 2014 ERCOT Hourly Load Data Archives `http://www.ercot.com/gridinfo/load/load_hist/`; c and d are chosen to closely match the wind output of the 2014 ERCOT Wind Integration Archives `http://www.ercot.com/gridinfo/generation/windintegration/2014`. Because the demand and wind output can vary significantly in some places, we also tested our algorithms in the case when $\sigma = 2$.
The results and analysis will be given in the next section.

6.3.1 Low variance case

In this section, we are going to analyze the CPU time, the percent error and the "Relative Efficiency" for different methods over a number of seasons when the demand and wind output are drawn from the lognormal distribution with the standard deviation of one. All of the numerical results for this section are given in Table 6.2.

Table 6.2: Numerical results when the demand and fuel cost are drawn from the lognormal distribution with $\sigma = 1$. The running time is rounded to minutes.

	CPU time (minutes)				% error of objective value				Relative Efficiency			
Number of time periods (seasons)	2	4	6	8	2	4	6	8	2	4	6	8
Uncluster	38	172	401	670	2.23	3.06	4.27	5.46	85.63	525.10	1713.76	3656.02
Uncluster + MCMC	34	149	348	559	1.97	2.83	3.46	4.58	66.55	420.54	1203.22	2558.92
Tech	7	44	119	175	3.08	4.58	6.21	7.03	22.54	203.35	737.75	1231.66
Tech + MCMC	5	42	112	166	3.04	4.06	5.74	6.89	15.91	169.57	644.03	1141.79
Tech + Size	14	73	175	290	2.77	3.83	4.97	6.52	38.23	280.36	867.76	1893.41
Tech + Size + MCMC	10	55	147	219	2.64	3.74	4.58	6.07	26.58	204.64	672.65	1331.56

Firstly, Figure 6.3 shows the CPU time of different methods for a number of seasons.

According to Figure 6.3, it can be seen that the order of the CPU time for different methods does not change over the number of season and it is shown in Table 6.3. Figure 6.3 and Table 6.3 show that the clustered formulations take significantly less amount of CPU time than the clustered formulation.

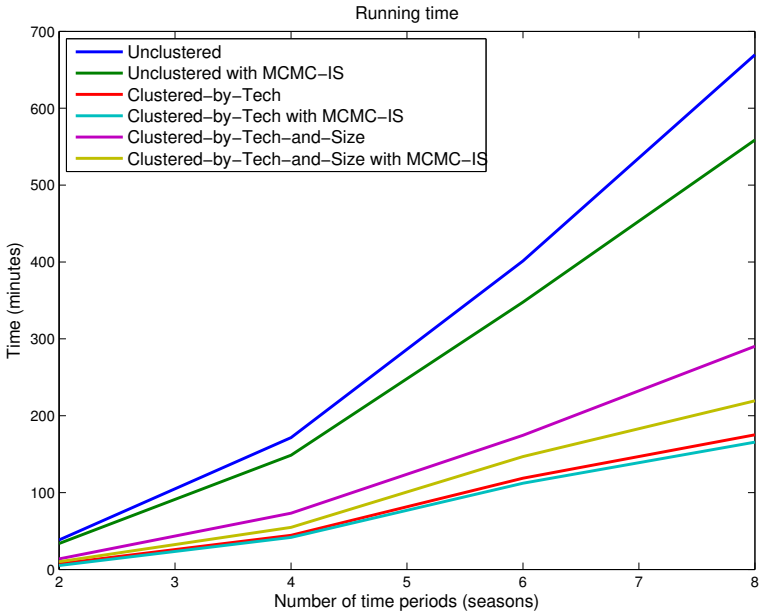


Figure 6.3: The CPU time of different methods for a number of seasons when the demand and wind output are drawn from the lognormal distribution with the standard deviation of one

1.	Clustered-by-Tech with MCMC-IS
2.	Clustered-by-Tech
3.	Clustered-by-Tech-and-Size with MCMC-IS
4.	Clustered-by-Tech-and-Size
5.	Unclustered with MCMC-IS
6.	Unclustered

Table 6.3: The order of the CPU time of different methods over a number of seasons (increasing order)

1.	Unclustered with MCMC-IS
2.	Unclustered
3.	Clustered-by-Tech-and-Size with MCMC-IS
4.	Clustered-by-Tech-and-Size
5.	Clustered-by-Tech with MCMC-IS
6.	Clustered-by-Tech

Table 6.4: The order of the % error of the objective cost for different methods over a number of seasons (increasing order)

The level of speed up is proportional to the aggregation level of the problem. For example: Since the size of the unclustered problem is about five times bigger than of the "Clustered-by-Tech" problem, it takes about four to five times greater than of the latter. The similar result can be seen when the CPU time of the "Unclustered" formulation is compared with the "Clustered-by-Tech-and-Size" one. More specifically, as the problem size of the former is about 2.5 times bigger than the size of the latter, it takes two to nearly three times longer for the "Unclustered" method to converge. Furthermore, the results show that the MCMC-IS algorithm takes less CPU time to solve a particular formulation than the SDDP algorithm. As the number of time periods increases, the advantage of the MCMC-IS algorithm becomes clearer (Figure 6.3). This implies that the cutting-planes construction for approximating the expected cost-to-go function in the MCMC-IS algorithm are more effective at every time period, and then accumulative over the number of time periods. The only case when the MCMC-IS algorithm does not show its clear advantage over the SDDP algorithm is in the "Clustered-by-Tech" formulation. This is because the size of the "Clustered-by-Tech" problem is relatively small, so the Monte Carlo method for generating samples in the SDDP algorithm is efficient enough for producing accurate results. On the other hand, the MCMC-IS algorithm takes some time for generating the MCMC samples and constructing the IS distribution to correct the bias. Therefore, the advantage of the MCMC-IS is easier to be realized in the large-scale problems such as the "Clustered-by-Tech-and-Size" and "Unclustered" formulation.
The % error of the objective cost for different methods for a number of seasons is shown in Figure 6.4.

According to Figure 6.4, the % error increases as the number of seasons increases. The order of the % error for different methods is shown in Table 6.4.

From Table 6.3 and Table 6.4, the results suggest that: Although the

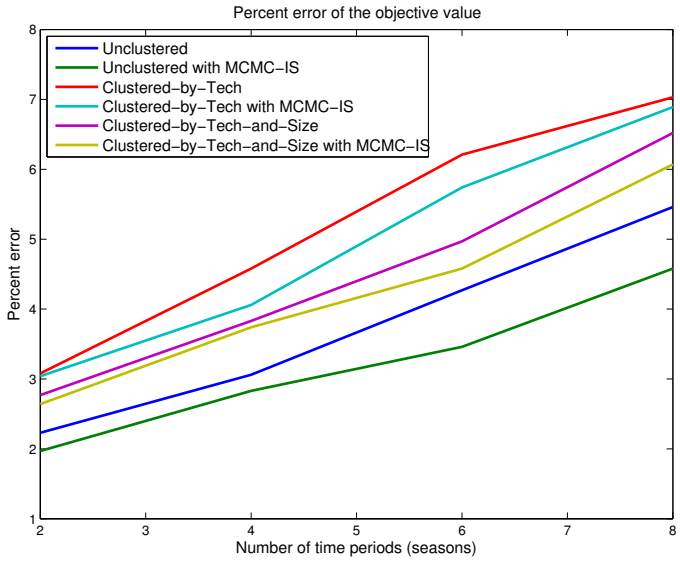


Figure 6.4: The % error of the objective cost for different methods for a number of seasons when the demand and wind output are drawn from the lognormal distribution with the standard deviation of one

aggregation reduces the CPU time to solve a problem, it increases the percentage error. This is because it assumes that all of the generators in the cluster are identical and hence, the clustered formulations will lose some details of every generator. Nevertheless, the % error increases very small, as shown in 6.2: All of the clustered methods have less than 2% increase in the % error relatively to the unclustered methods. For example: The % error increases from the "Unclustered" to "Clustered-by-Tech" method is 0.85%, 1.52%, 1.94% and 1.57% for the $2, 4, 6, 8-$seasons respectively, whereas the CPU time of the "Clustered-by-Tech" method is $5.2, 3.9, 3.4, 3.9$ times faster than of the former.
The "Relative Efficiency" that investigates the trade-off between the CPU time and the % error is shown in Figure 6.5.

According to Figure 6.5, the order of the "Relative Efficiency" of different methods over a number of seasons is shown in Table 6.5. According to Table 6.5, the top four methods use the clustered formulation while the bottom two positions belong to the unclustered formulations. This shows the advantage of the aggregation. For a given clustered formulation, the MCMC-IS algorithm always performs better than the SDDP algorithm. Therefore, the results indicate the advantage of MCMC-IS and clustering for solving this large scale problem.

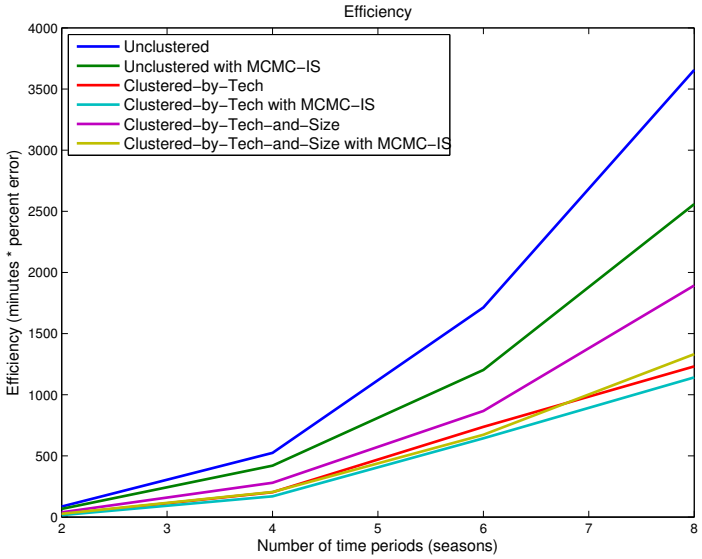


Figure 6.5: The "Relative Efficiency" of different methods over a number of seasons when the demand and wind output are drawn from the lognormal distribution with the standard deviation of one

1.	Clustered-by-Tech with MCMC-IS
2.	Clustered-by-Tech
3.	Clustered-by-Tech-and-Size with MCMC-IS
4.	Clustered-by-Tech-and-Size
5.	Unclustered with MCMC-IS
6.	Unclustered

Table 6.5: The order of the "Relative Efficiency" of different methods over a number of seasons. 1 = the best method. 6 = the worst method

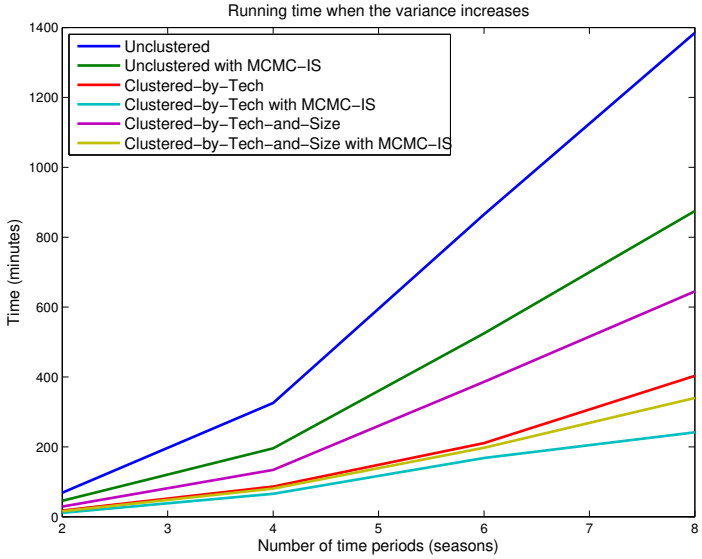


Figure 6.6: The CPU time of different methods for a number of seasons when the demand and wind output are drawn from the lognormal distribution with the standard deviation of two

6.3.2 High variance case

In this section, we are going to investigate the situation when the demand and wind output fluctuate strongly. Therefore, we will repeat the same procedure as in this section but the standard deviations of the lognormal distributions increases from one to two. The numerical results are given in Table 6.6.

Table 6.6: Numerical results when the demand and fuel cost are drawn from the lognormal distribution with $\sigma = 2$. The running time is rounded to minutes.

	Running time (minutes)				% error of objective value				Relative Efficiency			
Number of time periods (seasons)	2	4	6	8	2	4	6	8	2	4	6	8
Uncluster	69	326	865	1385	2.71	3.48	4.82	6.16	186.85	1133.09	4170.02	8531.39
Uncluster + MCMC	46	196	525	875	2.62	3.37	4.64	5.69	119.43	660.01	2434.14	4977.99
Tech	18	86	211	403	4.56	5.35	7.37	8.57	80.86	462.60	1552.12	3456.14
Tech + MCMC	11	66	168	242	3.97	4.78	6.38	7.83	44.53	314.21	1072.80	1892.51
Tech + Size	29	134	386	645	3.24	4.16	5.87	7.45	94.01	557.23	2265.62	4805.62
Tech + Size + MCMC	16	81	197	340	3.39	4.19	5.32	6.77	53.73	337.85	1049.55	2301.46

Figure 6.6 shows the CPU time of different methods for a number of seasons when the demand and wind output are drawn from the lognormal distribution with standard deviation of two. Figure 6.6 shows that all of the methods take longer than those of $\sigma = 1$. The order of the CPU time of different methods for a number of seasons is shown in Table 6.7. In comparison

1.	Clustered-by-Tech with MCMC-IS
2.	Clustered-by-Tech-and-Size with MCMC-IS
3.	Clustered-by-Tech
4.	Clustered-by-Tech-and-Size
5.	Unclustered with MCMC-IS
6.	Unclustered

Table 6.7: The order of the CPU time of different methods for a number of seasons (increasing order)

to Table 6.3, there is a swap between the position two and three. As a result, the two fastest methods are the "Clustered-by-Tech with MCMC-IS" and "Clustered-by-Tech-and-Size with MCMC-IS". It shows that even though the "Clustered-by-Tech" formulation has the smaller number of constraints and decision variables dimensions, the algorithm still takes a long time to converge. This can be explained by the fact that the convergence rate of the SDDP is proportional to the variance of the samples. In the SDDP algorithm, the samples are generated directly from the Monte Carlo method so their quality is influenced significantly when the standard deviation increases. On the other hand, the samples used in the MCMC-IS algorithm are generated from the *zero-variance* IS distribution so the MCMC-IS algorithm can perform robustly in many distributions Parpas et al. (2014). As a result, the MCMC-IS algorithm with the clustered formulation are the key for making the problem tractable when the uncertainties are drawn from the high-variance distribution.
The percentage error of the objective cost for different methods for a number of seasons is shown in Figure 6.7.

Similarly to the case of $\sigma = 1$, Figure 6.7 shows that the % error of the objective cost increases with the number of time periods. In comparison the the % error when $\sigma = 1$ (Table 6.2), the % error of the objective cost when $\sigma = 2$ (Table 6.6) increases by less than one percent for most of the methods and only one percent in the others.
The "Relative Efficiency" of different methods are shown in Figure 6.8.

According to Figure 6.8, the order of the "Relative Efficiency" of different methods for a number a seasons is shown in Table 6.8. In comparison to the previous case when $\sigma = 1$ (Table 6.5), Table 6.8 shows that the bottom three methods are unchanged; there is, however, a big change in the top three positions. When $\sigma = 1$, the "Clustered-by-Tech with MCMC-IS", "Clustered-by-Tech" and "Clustered-by-Tech-and-Size with MCMC-IS"

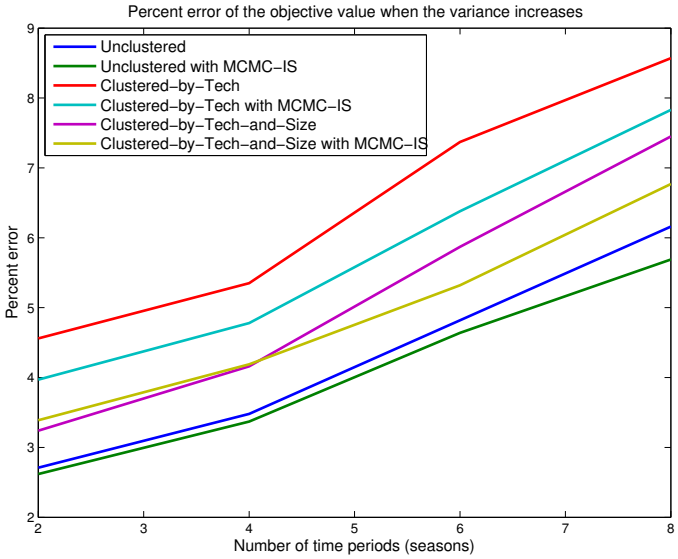


Figure 6.7: The % error of the objective cost for different methods when the demand and wind output are drawn from the lognormal distribution with standard deviation of two

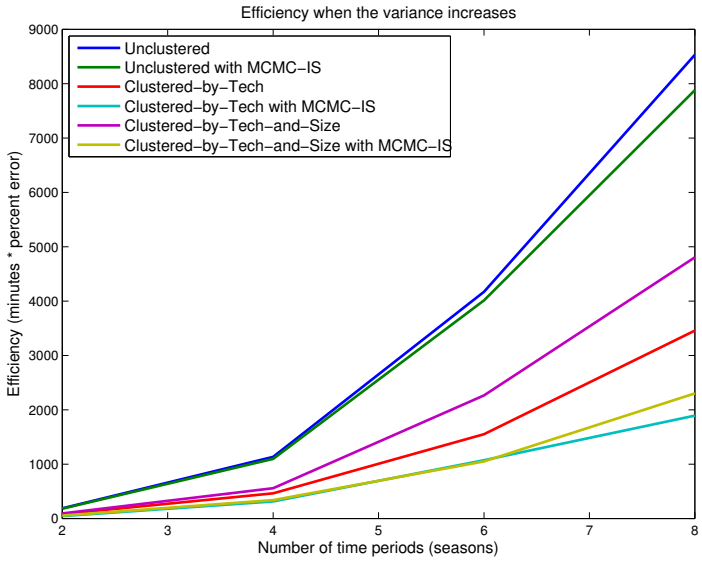


Figure 6.8: The efficiency of six methods when the demand and wind output are drawn from the Normal distribution with high standard deviation

1.	Clustered-by-Tech with MCMC-IS
2.	Clustered-by-Tech-and-Size with MCMC-IS
3.	Clustered-by-Tech
4.	Clustered-by-Tech-and-Size
5.	Unclustered with MCMC-IS
6.	Unclustered

Table 6.8: The order of the CPU time of different methods for a number of seasons (increasing order)

1.	Unclustered with MCMC-IS
2.	Unclustered
3.	Clustered-by-Tech-and-Size with MCMC-IS
4.	Clustered-by-Tech-and-Size
5.	Clustered-by-Tech with MCMC-IS
6.	Clustered-by-Tech

Table 6.9: The % error of the optimal decision variables for different methods over a number of seasons (increasing order)

methods have nearly equal performances. When $\sigma = 2$, the "Clustered-by-Tech with MCMC-IS" and "Clustered-by-Tech-and-Size with MCMC-IS" methods show their clear advantages over the "Clustered-by-Tech" method. As a result, it suggests that the combination of the MCMC-IS algorithm and the clustered formulation is key for solving our integrated model efficiently when the random variables are drawn from a high-variance distribution. In the next section, we are going to investigate the optimal decision variables of different methods.

Figure 6.9 and Figure 6.10 show the % error of the optimal decision variables for different methods over a number of seasons. They are summarized in Table 6.9. Table 6.9 shows that the "Unclustered with MCMC-IS" method gives the lowest % error of the optimal decision variables. However, as shown in Figure 6.3 and Figure 6.6, this method is very computationally expensive for a large number of time periods. Since the generators are built to last for several years or decades, it is important to include as large number of time periods as possible in order to take into account the changes in the future demand. In the next section, we are going to investigate the quality of the optimal decision variables obtained by different clustered methods and then compare them with the "Unclustered with MCMC-IS" method.

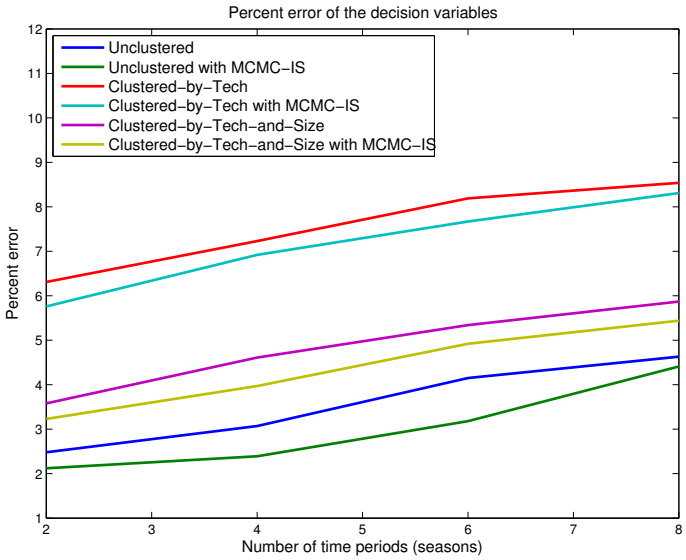


Figure 6.9: The % error of the optimal decision variables for different methods over a number of seasons when the demand and wind output are drawn form the lognormal distribution with the standard deviation of one

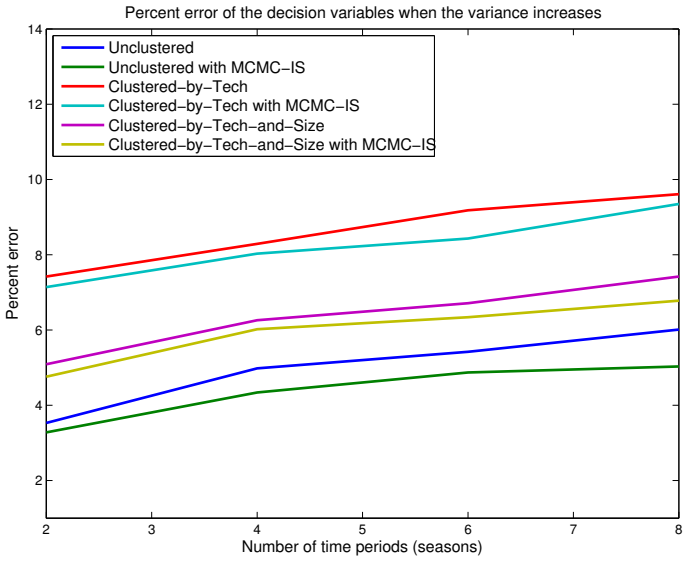


Figure 6.10: The % error of the optimal decision variables for different methods over a number of seasons when the demand and wind output are drawn from the lognormal distribution with the standard deviation of two

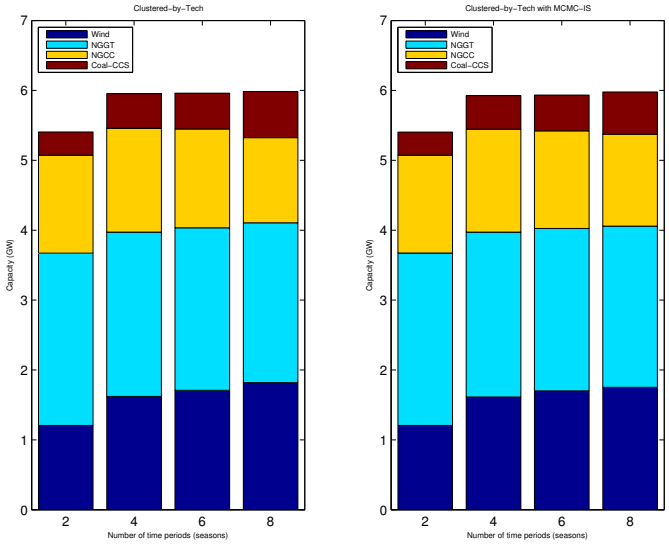


Figure 6.11: Optimal decision variables for the "Clustered-by-Tech" method

6.3.3 The optimal capacity expansion decisions

In this section, we are going to analyze the optimal capacity expansion decisions given by different clustered methods and then compare them with the "Unclustered with MCMC-IS" method. The optimal decision variables for the "Clustered-by-Tech" method is given in Figure 6.11, and for the "Clustered-by-Tech-and-Size" method is given in Figure 6.12 in the case $\sigma = 1$. The optimal decision variables for the "Clustered-by-Tech" method is given in Figure 6.13 , and for the "Clustered-by-Tech-and-Size" method is given in Figure 6.14 in the case $\sigma = 2$.

As shown in Figure 6.11 and Figure 6.12, the total new installed capacity increases as the number of time periods increases from two to four seasons. After that, it remains relatively the same for the four, six and eight seasons. This is because: In the first two seasons, the demand has not reached its peak of the year yet. The peak demand happens during the last two seasons of the first year and therefore, it can only be considered if the problem has the number of time periods up to four seasons or above. Therefore, this is one example to show that the problem may give sub-optimal decision variables if it does not include a large number of time periods. Since the demand in the second year is assumed to remain the same as in the first year, the new installed capacity for the six and eight seasons does not change much after the first year.

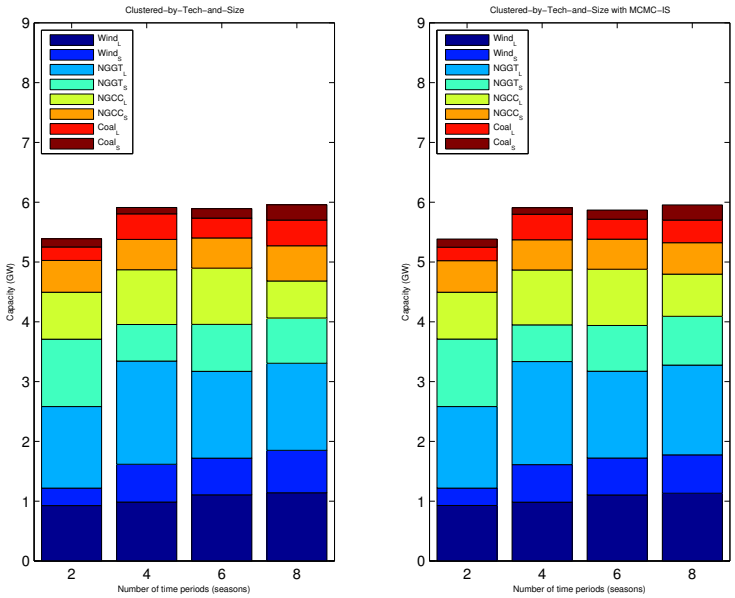


Figure 6.12: Optimal decision variables for the "Clustered-by-Tech-and-Size" method

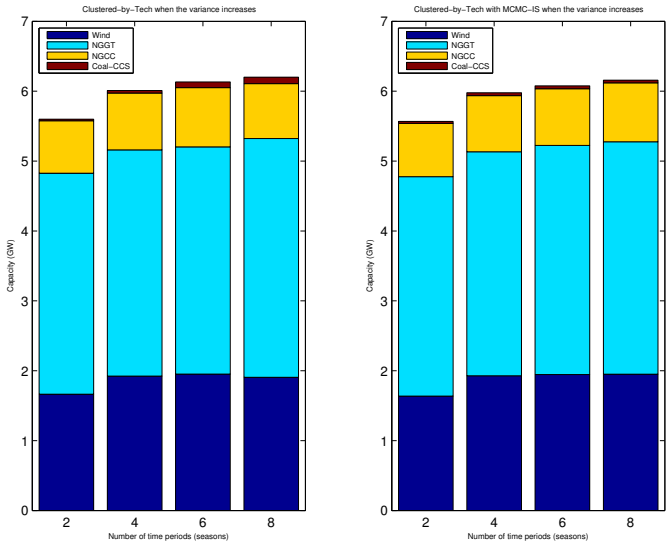


Figure 6.13: Optimal decision variables for the "Clustered-by-Tech" method when the variance increases

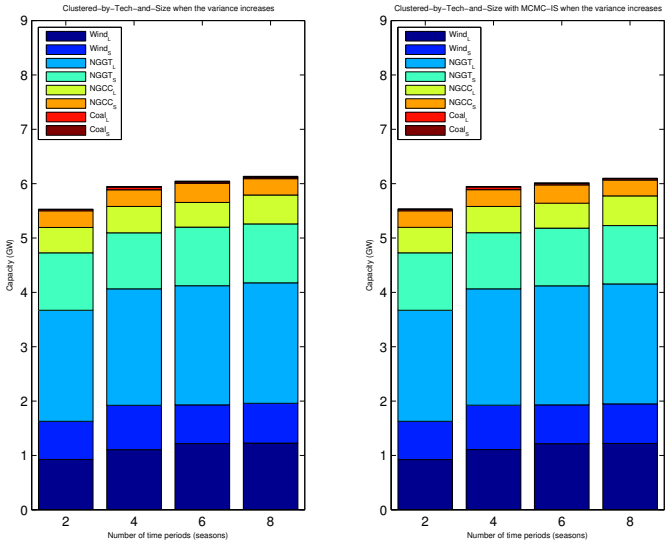


Figure 6.14: Optimal decision variables for the "Clustered-by-Tech-and-Size" method when the variance increases

Moreover, the results show that the wind capacity increases as the problem increases from two to four seasons. This can be explained by the fact that the wind output during the last two seasons of the year is higher than of the first two seasons. Hence, more wind capacity needs to build to ensure that at least 20% of the power generation come from the wind output. Besides, the capacity for "Coal-CCS" also increases as the problem increases from two to four seasons. This is because the demand during the last two seasons increases. The total capacity thus has to increase. In this case, the capacity expansion of Coal-CCS will keep the operating cost low while the NGGT are able to provide enough flexibility to respond to the uncertainties in the demand and wind output.
Figure 6.13 and Figure 6.14 show that the capacity for Coal-CCS reduces significantly when $\sigma = 2$. Instead, a large capacity of NGGT is built to increase the system's operational flexbility in order to cope with the high fluctuation of the wind output. In comparison to the case of $\sigma = 1$, the capacity for NGCC decreases; however, unlike the Coal-CCS, NGCC still play an important role as they have relatively low operating cost with medium-to-high operational flexibility. In addition, the installed capacity for wind increases as the wind output can go up significantly due to the high standard deviation, and the system has to be ready for capturing as much wind output as possible.

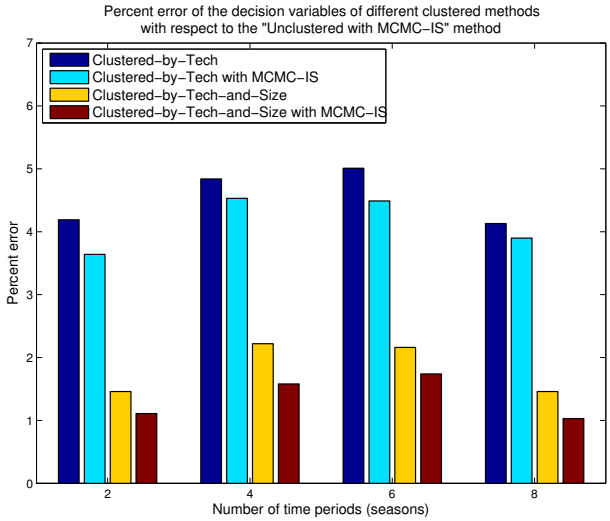


Figure 6.15: Difference between the MSE of the optimal decision variables and the "true" solution given by the "Unclustered with MCMC-IS" method

In summary, when the demand and wind output vary significantly, the optimal capacity expansion decisions will change accordingly. They are shifting towards using more highly flexible generators such as NGGT while reducing the capacity of the less flexible technologies (i.e. Coal-CCS). More wind capacity is also required to leverage the wind output when there is strong wind. The next step is to compare the quality of the optimal decision variables of different clustered methods with the "Unclustered with MCMC-IS" method, which achieves the lowest % error in the optimal decision variables but may suffer the curse of complexity. Figure 6.15 shows the % error of the optimal decision variables of different clustered methods relatively to the "Unclustered with MCMC-IS" method when the demand and wind output are drawn from the lognormal distribution with the standard deviation of one. Figure 6.16 shows the % error of the optimal decision variables of different clustered methods relatively to the "Unclustered with MCMC-IS" method when the demand and wind output are drawn from the lognormal distribution with the standard deviation of two.

Figure 6.15 and Figure 6.16 show that the optimal decision variables in the "Clustered-by-Tech-and-Size" and "Clustered-by-Tech-and-Size with MCMC-IS" have about 1.5% error, whereas the percent error in the optimal decision variables given by the "Clustered-by-Tech" and "Clustered-by-Tech with MCMC-IS" is about 3.7% to 4.5%.

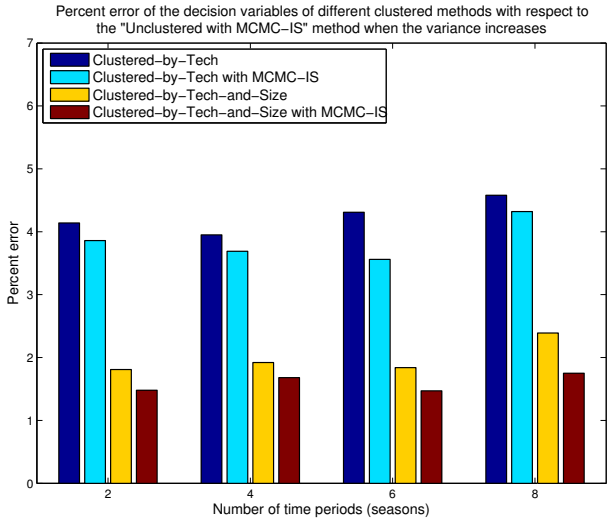


Figure 6.16: Difference between the MSE of the optimal decision variables and the "true" solution given by the "Unclustered with MCMC-IS" method

6.4 Summary

We have presented a multistage stochastic program that integrates the unit-commitment, the maintenance scheduling and the capacity expansion models into a single framework. We then proposed different ways of aggregation and two different optimization algorithms to solve this problem. The numerical experiments show that the MCMC-IS algorithm in combination with the clustered formulation are key for solving it efficently. When the demand and wind output fluctuate strongly, the "Clustered-by-Tech with MCMC-IS" method can speed up to almost six times the "Unclustered" method while the error only increases by one percent. On the other hand, the "Clustered-by-Tech-and-Size with MCMC-IS" method can speed up by four times the "Unclustered" method for almost zero increase in the percent error. Moreover, the results show that as the demand and wind output fluctuate strongly, the power system has to increase the capacity of the highly flexible generators such as NGGT while relying less on the less flexible generators such as Coal-CCS.

Chapter 7

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 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 thesis 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 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 thesis 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.

We have also proved the convergence of the proposed algorithm. We will investigate the complexity of the algorithm in the future. We then used the proposed algorithm to solve the capacity expansion planning problem in the electric power industry. This problem is a very large scale optimization problem, which includes another two very challenging problems: The unit commitment problem and the maintenance scheduling problem. Therefore, we had to reduce the number of dimensions and the number of constraints of the problems before trying to solve it. We therefore used different types of clustering methods to group power generators of similar characteristics together. This approach helps to reduce the problem size; however, the problem is still intractable, especially when the model includes a large number of time periods in order to seek for better long-term investment decisions or when the uncertainties such as the demand and wind output vary significantly. Then the numerical experiments showed that the MCMC-IS algorithm in combination with the clustered formulation are key for solving it efficently and accurately. There is a tradeoff between the computational time and the accuracy of the solutions for different aggregation levels. However, this tradeoff can be kept minimal if we select the right clustering features. For example: When the electric demand and wind output fluctuate strongly, our results show that the "Clustered-by-Tech with MCMC-IS" method can speed up to almost six times the "Unclustered with SDDP" method while the error only increases by one percent. On the other hand, the "Clustered-by-Tech-and-Size with MCMC-IS" method can speed up by four times the "Unclustered" method for almost zero increase in the percent error. In terms of the optimal solutions, the results showed that as the demand and wind output fluctuate strongly, the power system has to increase the capacity of the highly flexible generators such as NGGT while relying less on the less flexible generators such as Coal-CCS.

Chapter 8

Appendix

Appendix A: Description of the Test Problems from Section 5.3

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 lognormal: $d_2 = 1700 \times \exp(0.1 \times \xi_2)$, 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 is obtained 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 remaining 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 increased 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. 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: $w(\xi) = \exp(\frac{\xi^2}{2} - \frac{(\xi+3)^2}{8}) + \exp(\frac{\xi^2}{2} - \frac{(\xi+1)^2}{8})$.

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 5.3

Table 8.1: 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($\tilde{z}$) (%)			Variance($\tilde{z}$) (%)			# 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 8.2: Comparison of performance of MC, QMC and the proposed algorithm (MCMC) when the random variables are drawn from the distribution with standard deviation of 2

Problem	# Samples (N) per period	MSE($\tilde{z}$) (%)			Variance($\tilde{z}$) (%)			# of Iterations			Runtime (s)		
		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 8.3: 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($\tilde{z}$) (%)			Variance($\tilde{z}$) (%)			# 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

Table 8.4: Average number of rejections when generating 3000 samples (M_r)

Problem	$\sigma = 1$	$\sigma = 2$	Rare-event
AOS	3849	6287	14154
FP	7683	9974	17408
EI	8952	12889	23785
SCO	5086	7514	15987
FPM	12463	15518	20584
DBCP	4653	6782	10562
EEP	13949	18318	28185
TNP	10761	15951	32714
BIP	9605	12789	35314

Bibliography

Abilock, H., C. Bergstrom, J. Brady. 1979. Markal: a multiperiod, linear programming model for energy systems analysis. *Technical Report* .

Ariyawansa, K., A Felt. 2004. On a new Collection of Stochastic Linear Programming Test Problems. *INFORMS Journal on Computing* **16**(3) 291–299.

Asmussen, S. 2003. *Applied Probability and Queues*. Springer.

Asmussen, S., P.W. Glynn. 2007. *Stochastic Simulation: Algorithms and Analysis*, *Stochastic Modelling and Applied Probability*, vol. 57. Springer, New York.

Athreya, K.B., G.S. Atuncar. 1998. Kernel estimation for real-valued markov chains. *The Indian Journal of Statistics* **60** 1–17.

Athreya, K.B., H. Doss, J. Sethuraman. 1996. On the convergence of the markov chain simulation method. *The Annals of Statistics* **24** 69–100.

Athreya, K.B., P. Ney. 1978. A new approach to the limit theory of recurrent markov chains. *Transactions of the American Mathematical Society* **245** 493–501.

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

Bastin, F., C. Cirillo, P. Toint. 2006. Convergence theory for nonconvex stochastic programming with an application to mixed logit. *Mathematical Programming* **108** 207–234.

Beale, E.M.L. 1955. On Minimizing A Convex Function Subject to Linear Inequalities. *Journal of the Royal Statistical Society* **17** 173–184.

Benders, J.F. 1962. Partitioning procedures for solving mixed-variables programming problems. *Numerische Mathematik* .

Birge, J.R., F. Louveaux. 2011. *Introduction to Stochastic Programming*. Springer Verlag.

Birge, J.R., F.V. Louveaux. 1988. A multicut algorithm for two-stage stochastic linear programs. *European Journal of Operational Research* **34** 384–392.

Bishop, C.M. 2006. *Pattern Recognition and Machine Learning*. Springer.

Botterud, A., M.D. Ilic, I. Wangensteen. 2005. Optimal investments in power gen-

eration under centralized and decentralized decision making. *IEEE Transactions on Power Systems* .

Bucklew, J.A. 2004. *Introduction to Rare Event Simulation*. Springer Series in Statistics, Springer-Verlag, New York.

Celebi, M.E., H.A. Kingravi, P.A. Vela. 2013. Expert systems with applications. *A comparative study of efficient initialization methods for the k-means clustering algorithm* .

Chao, H. 2010. Nyiso Reliability & Economic Planning Process. *FERC Technical Conference on Planning Models and Software* .

Chattopadhyay, D., K. Bhattacharya, J. Parikh. 1995. A system approach to least-cost maintenance scheduling for an interconnected power system. *IEEE Transactions on Power Systems* **10** 2002–2007.

Chattopadhyay, D., K. Bhattacharya, J. Parikh. 1998. A practical maintenance scheduling program: Mathematical model and case study. *IEEE Transactions on Power Systems* **13** 1475–1480.

Chib, S., E. Greenberg. 1995. Understanding the metropolis-hastings algorithm. *The American Statistician* **49** 327–335.

Comaniciu, D., P. Meer. 2002. Mean shift: A robust approach toward feature space analysis. *IEEE Transactions on Pattern Analysis and Machine Intelligence* .

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

Dantzig, G.B. 1955. Linear Programming under Uncertainty. *Management Science* **1** 197–206.

Dantzig, G.B., P.W. Glynn. 1990. Parallel Processors for Planning Under Uncertainty. *Annals of Operations Research* **22**(1) 1–21.

Dantzig, G.B., P. Wolfe. 1960. Decomposition principle for linear programs. *Operations Research* **8** 101–111.

Dempster, M.A.H. 1988. On stochastic programming ii: dynamic problems under risk. *Stochastics* **25** 15–42.

Dick, J., F. Pillichshammer. 2010. *Digital Nets and Sequences*. Cambridge University Press.

Drew, S.S. 2007. Quasi-monte carlo methods for stochastic programming. *Ph.D. thesis, Northwestern University* .

Drew, S.S., T. Homem-de Mello. 2006. Quasi-Monte Carlo Strategies for StochasticOptimization. *Proceedings of the 38th conference on Winter simulation*. Winter Simulation Conference, 774–782.

Drew, S.S., T. Homem-de Mello. 2012. Some Large Deviations Results for Latin Hypercube Sampling. *Simulation Conference* **14** 203–232.

Dupačová, J., N. Gröwe-Kuska, W. Römisch. 2003. Scenario Reduction in Stochastic Programming. *Mathematical Programming* **95**(3) 493–511.

European Commission. 2011. Energy roadmap 2050 .

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

Gassman, H.I. 1990. Mslip: A computer code for the multistage stochastic linear programming problem. *Mathematical Programming* **47** 407–423.

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

Gilks, W.R., S. Richardson, D.J. Spiegelhalter. 1996. *Markov chain Monte Carlo in practice*. Chapman and Hall.

Gondzio, J., A. Grothey. 2006. Direct Solution of Linear Systems of size 10^9 arising in Optimization with Interior Point Methods. *Parallel Processing and Applied Mathematics*. Springer, 513–525.

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

Hastings, W.K. 1970. Monte carlo sampling methods using markov chains and their applications. *Biometrika* **57** 97–109.

Hestenes, M.R. 1969. Multiplier and gradient methods. *Journal of Optimization Theory and Applications* **4** 303–320.

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.

Higle, J.L., S. Sen. 2013. *Stochastic Decomposition: A statistical Method for Large Scale Stochastic Linear Programming (Nonconvex Optimization and Its Applications)*. Springer.

Hobbs, B.F., M.H. Rothkopf, R.P. O'Neil, H. Chao. 2001. *The Next Generation of Electric Power Unit Commitment Models*. Springer.

Hodges, J.L., E.L. Lehmann. 1956. The efficiency of some nonparametric competitors of the t-test. *The Annals of Mathematical Statistics* **27** 324–335.

Homem-de Mello, T. 2008. On Rates of Convergence for Stochastic Optimization Problems under non-iid Sampling. *SIAM Journal on Optimization* **19** 524–551.

Homem-de Mello, T., G. Bayraksan. 2014. Monte carlo sampling-based methods for stochastic optimization. *Surveys in Operations Research and Management Science* .

Homem-de Mello, T., V.L.D Matos, E.C. 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.

Infanger, G. 1992. Monte Carlo (Importance) Sampling within a Benders Decomposition algorithm for Stochastic Linear Programs. *Annals of Operations Research* **39**(1) 69–95.

Intergovernmental Panel on Climate Change. 2014. Climate change 2014: Synthesis report. contribution of working groups i, ii and iii to the fifth assessment report of the intergovernmental panel on climate change .

International Energy Agency. 2008. Empowering variable renewables - options for flexible electricity systems .

Jones, M.C., J.S. Marron, S.J. Sheather. 1996. A brief survey of bandwidth selection for density estimation. *Journal of the American Statistical Association* **91** 401–407.

Kalagnanam, J., U. Diwekar. 1997. An efficient sampling technique for off-line quality control. *Technometrics* **39** 308–319.

Kleywegt, A.J., A. Shapiro, T. Homem-de Mello. 2001. The sample average approximation method for stochastic discrete optimization. *SIAM Journal on Optimization* **12** 479–502.

Koivu, M. 2005. Variance Reduction in Sample Approximations of Stochastic Programs. *Mathematical Programming* **103**(3) 463–485.

Kuhn, D., W. Wiesemann, A. Georghiou. 2011. Primal and dual linear decision rules in stochastic and robust optimization. *Mathematical Programming* **130** 177–209.

Lannoye, E. 2011. The role of power system flexibility in generation planning. *Power and Energy Society General Meeting* .

Lemieux, C. 2009. *Monte Carlo and Quasi-Monte Carlo Sampling*. Springer.

Li, T., M. Shahidehpour. 2005. Priced-based unit commitment: A case of lagrangian relaxation versus mixed integer programming. *IEEE Transactions on Power Systems* **20** 2015–2025.

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

Lubin, M., C.G Petra, M. Anitescu, V. Zavala. 2011. Scalable Stochastic Optimization of Complex Energy Systems. *High Performance Computing, Networking, Storage and Analysis (SC), 2011 International Conference for*. IEEE, 1–10.

McKay, M.D., R.J. Beckman, W.J. Conover. 1979. A comparison of three methods for selecting values of input variables in the analysis of output from a computer code. *Technometrics* **21** 239–245.

Metropolis, N., A.W. Rosenbluth, M.N. Rosenbluth, A.H. Teller, E. Teller. 1953. Equation of state calculations by fast computing machines. *Journal of Chemical Physics* **21** 1087–1092.

National Renewable Energy Laboratory. 2012. Renewable electricity futures report .

Niederreiter, H. 1992. *Random Number Generation and Quasi-Monte Carlo Methods*. Society for Industrial and Applied Mathematics.

North American Electric Reliability Corporation. 2009. Accommodating high levels of variable generation .

Northwest Power and Conservation Council. 2010. Sixth northwest conservation and electric power plan .

Owen, A.B. 1992. A Central Limit Theorem for Latin Hypercube Sampling. *Journal of the Royal Statistical Society* **54** 541–551.

Owen, A.B. 1998. Latin supercube sampling for very high-dimensional simulations. *ACM Transactions on Modeling and Computer Simulation* **8** 71–102.

Owen, A.B. 2003. The dimension distribution and quadrature test functions. *Statistica Sinica* **13** 1–17.

Padhy, N.P. 2004. Unit commitment - a bibliographical survey. *IEEE Transactions on Power Systems* .

Palmintier, B.S. 2013. Incorporating operational flexibility into electric generation planning: Impacts and methods for system design and plicy analysis. Master's thesis, Massachusetts Institute of Technology.

Parpas, P., B. Rustem. 2007. Computational assessment of nested benders and augmented lagrangian decomposition for mean-variance multistage stochastic problems. *INFORMS Journal on Computing* **19** 239–247.

Parpas, P., B. Ustun, M. Webster, Q.K. Tran. 2014. Importance sampling in stochastic programming: A markov chain monte carlo approach. *INFORMS Journal on Computing* **27** 358–377.

Parpas, P., M. Webster. 2013. A stochastic minimum principle and an adaptive pathwise algorithm for stochastic optimal control. *Automatica* **49** 1663–1671.

Parzen, E. 1962. On estimation of a probability density function and mode. *Annals of Mathematical Statistics* **33** 1065–1076.

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.

Philpott, A.B., Z. Guan. 2008. On the convergence of stochastic dual dynamic programming and related methods. *Operations Research Letter* .

Powell, M.J.D. 1969. A method for nonlinear constraints in minimization problems. *Optimization* 283–298.

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

Prekopa, A. 1995. *Stochastic Programming*. Springer.

Rao, B.L.S.P. 2014. *Nonparametric Functional Estimation*. Elsevier Science.

Ravindran, A. Ravi, ed. 2008. *Operations Research and Management Science Handbook*. Operations Research Series, CRC Press, Boca Raton, FL.

Robert, C.P., G. Casella. 2013. *Monte Carlo Statistical Methods*. Springer.

Roberts, G.O., J.S. Rosenthal. 2006. Harris recurrence of metropolis-within-gibbs and trans-dimensional markov chains. *The Annals of Applied Probability* **16** 2123–2139.

Rockafellar, R.T., R.J.B. Wets. 1991. Scenarios and Policy Aggregation in Optimization Under Uncertainty. *Mathematics of Operations Research* 119–147.

Ruszczynski, A. 1986. A regularized decomposition for minimizing a sum of polyhedral functions. *Mathematical Programming* **35** 309–333.

Scott, D. 1992. *Multivariate Density Estimation: Theory, Practice, and Visualization*. 1st ed. Wiley.

Scott, D.W. 2015. *Multivariate Density Estimation: Theory, Practice and Visualization, Second Edition*. WileySeries in Probability and Statistics.

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 and Applied Mathematics.

Shapiro, A., T. Homem-de Mello, J.C. Kim. 2002. Conditioning of convex piecewise linear stochastic programs. *Mathematical Programming* **94** 1–19.

Shapiro, A., A. Nemirovski. 2005. On complexity of stochastic programming problems. *Continuous Optimization* **99** 111–146.

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

Silverman, B.W. 1998. *Density Estimation for Statistics and Data Analysis*. CRC Press.

Tierney, L. 1994. Markov chains for exploring posterior distributions. *The Annals of Statistics* **22** 1701–1762.

U.S. Energy Information Administration. 2010. Updated capital cost estimates for electricity generation plants .

U.S. Environmental Protection Agency. 2010. The emissions and generation resource integrated database (egrid) .

Weber, C. 2005. *Uncertainty in the electric power industry - Methods and Models for Decision Support*. Springer.

Wiser, R., G. Barbose. 2008. Renewables portfolio standards in the united states - a status report with data through 2007. *Technical report LBNL-154E, Lawrence Berkeley National Laboratory* **13**.