12-2009

Polynomial approximation method for stochastic programming.

Dongxue Ma 1986-
University of Louisville

POLYNOMIAL APPROXIMATION METHOD FOR STOCHASTIC PROGRAMMING

By

Dongxue Ma

B.A., Beijing University of Posts and Telecommunications, 2008

A Thesis

Submitted to the Faculty of the Graduate School of the University of Louisville

in Partial Fulfillment of the Requirements for the degree of

Master of Science

Department of Industrial Engineering

UNIVERSITY OF LOUISVILLE

Louisville, Kentucky

December 2009

Polynomial Approximation Method for Stochastic Programming

By

Dongxue Ma

B.A., Beijing University of Posts and Telecommunications, 2008

A Thesis Approved on

August 30th, 2009

by the following Thesis Committee:

Thesis Director

DEDICATION

This thesis is dedicated to my parents

Mr. Jing Ma

and

Mrs. Lihua Yang

who have given me invaluable educational opportunities.

ACKNOWLEDGMENT

I would like to thank my advisor, Dr. Lijian Chen, for his guidance and patience. I would also like to express my thanks to my parents, for their understanding and support during those times when there was no light at the end of anything. I would also like to thank Dr.Gail W. DePuy and Dr. Tim Hardin who gave me many helpful suggestions. Finally, I would like to thank Dr. Stephen Boyed, School of Engineering, Stanford University who developed interesting, unique, and resourceful optimization material in both publication and online. CVX software package is quite helpful in solving the norm approximation model.

ABSTRACT

POLYNOMIAL APPROXIMATION METHOD FOR STOCHASTIC PROGRAMMING

Dongxue Ma

October 2nd, 2009

Two stage stochastic programming is an important part in the whole area of stochastic programming, and is widely spread in multiple disciplines, such as financial management, risk management, and logistics. The two stage stochastic programming is a natural extension of linear programming by incorporating uncertainty into the model. This thesis solves the two stage stochastic programming using a novel approach. For most two stage stochastic programming model instances, both the objective function and constraints are convex but non-differentiable, e.g. piecewise-linear, and thereby solved by the first gradient-type methods. When encountering large scale problems, the performance of known methods, such as the stochastic decomposition (SD) and stochastic approximation (SA), is poor in practice. This thesis replaces the objective function and constraints with their polynomial approximations. That is becauce polynomial counterpart has the following benefits:first, the polynomial approximation will preserve the convexity; Second, the polynomial approximation will *uniformly* converge to the original objective/constraints with arbitrary accuracy; and third, the polynomial approximation will not only provide good estimation on the original objectives/functions but also their gradients/sub-gradients. All these features enable us to apply convex optimization techniques for large scale problems. Hence, the thesis applies SAA, polynomial approximation method and then steepest descent method in combination to solve the large-scale problems effectively and efficiently.

TABLE OF CONTENTS

LIST OF TABLES

LIST OF FIGURES

CHAPTER 1

INTRODUCTION

1.1. THESIS STRUCTURE

The thesis is organized in the following way. In Chapter 1, it is carefully demonstrated how to incorporate random variables into a linear program and the process of reaching a general formula of two stage stochastic programming. Considering the wide applications of stochastic programming, several classic stochastic models which adjusts to typical application are also introduced. Thereafter, this thesis shows that the computational complexity is often a problem for the resulting model. Actually, the model scale grows exponentially. Prior to presenting the detailed solution, a thorough literature review is conducted to summarize articles most useful and classical approaches for stochastic programming.

In Chapter 2, the thesis presents the typical model of two stage stochastic programming and its derivatives. The polynomial approximation is then constructed with mathematical proof to show the valued properties in both theory and practice. For a practical method, the thesis provides the necessary polynomial degree and the necessary number of replications required. In Chapter 3, a large scale resource distribution problem is solved and the numerical result is presented to conclude our research.

1.2. MODELING UNCERTAINTY

Optimization is an important tool for most industries. Airline companies use optimization to schedule crews and aircraft. Investors use optimization to minimize cost and risk, which achieves a high rate of return. There is no doubt that optimization is quite critical in decision making. Optimization, also called mathematical programming, chooses the best solution from some set of feasible alternatives. A typical optimization problem should contains a set of constraints and an objective which is often used as a quantitative measurement of the performance of the system under study.

The first known optimization technique, steepest descent, which dates to Carl Friedrich Gauss. After World War II, the modern optimization techniques emerged by the introduction of linear programming by George Dantzig in the 1940s. Over the past several decades, linear programming has become a fundamental planning tool. It is widely applied in engineering, business, economics, environmental studies and other disciplines. For example, the job assignment, transportation control, network volume control, resource allocation, and scheduling problems can be solved easily with the application of linear programming.

The general form of all kinds of linear programming models can be written as follows:

$$\min \{c'x | Ax \leq b\} \tag{1.1}$$

where x represents the vector of variables (to be determined), while c and b are vectors of (known) coefficients and A is a matrix of coefficients. The expression to be maximized or minimized is called the objective function ($c'x$ in this case). The equations $Ax \leq b$ are the constraints which specify a convex polyhedron over which the objective function is to be optimized.

Nevertheless, linear programming has its own drawbacks and limitations. First of all, linear programming is applicable only to problems where the constraints and objective function are linear, i.e., where they can be expressed as equations that represent straight lines. In real life situations, when constraints or objective functions are not linear, this technique cannot be used. Secondly, the simplex method is a well-known method to solve linear programing problems most of the time. Although this algorithm is quite efficient in practice and can guarantee to find the global optimum if certain precautions against cycling are taken, it has poor worst-case behavior: it is possible to construct a linear programming problem for which the simplex method takes a number of steps exponential in the problem size.

In fact, for some time it was not known whether the linear programming problem was solvable in polynomial time. But with the introduction of the ellipsoid method by *Leonid Khachiyan* in 1979, this issue was solved. Although it is not a practical method, it provides insightful views on the possibilities of developing efficient polynomial algorithms by the interior point method. In 1984, *N. Karmakar* proposed a new interior point projective method for linear programing which not only improved *Khachiyan*'s theoretical worst-case polynomial bound, but also promised dramatic practical performance improvements over the simplex method.

The third limitation of linear programming problems is the optimal solution may become severely infeasible if the nominal data is slightly changed. The general form of linear programming model is provided earlier (1.1). If some coefficients of any constrain are slightly changed, then the optimal solution will become infeasible. This phenomenon has been demonstrated by studying `PILLOT4` from the well-known `NETLIB` collection, see [4]. It is a linear programming with 1000 variables and 410 constraints; One of the constraints

(#372) is

$$\begin{aligned} a^T x = & 15.79081 x_{826} - 8.598819 x_{827} - 1.88789 x_{828} 1.362417 x_{829} \\ & 1.526049 x_{830} - 0.031883 x_{849} - 28.725555 x_{850} - 10.792065 x_{851} \\ & 0.19004 x_{852} - 2.757176 x_{853} - 12.290832 x_{854} + 717.562256 x_{855} \\ & 0.057865 x_{856} - 3.785417 x_{857} - 78.30661 x_{858} - 122.163055 x_{859} \\ & 6.46609 x_{860} - 0.48371 x_{861} - 0.615264 x_{862} - 1.353783 x_{863} \\ & 84.644257 x_{864} - 122.459045 x_{865} - 43.15593 x_{866} - 1.712592 x_{870} \\ & 0.401597 x_{871} + x_{880} - 0.946049 x_{898} - 0.946049 x_{916} \geq b = 23.387405 \end{aligned} \tag{1.2}$$

After doing some experiments, they found that quite small (just 0.1%) perturbations of "obviously uncertain" data coefficients can make the "normal" optimal solution x^* heavily infeasible and thus practically meaningless.

Last but not least, linear programming problems assume that all model parameters are known with certainty. This is hardly true for real world problems. The most likely case is that we only know *partial information.* When some parameter is not known, the linear programming model actually becomes a stochastic programming model and traditional solutions become invalid. We mainly focus on introducing an approach based on a probabilistic pattern of the uncertain data. In other words, this method considers the probabilities of the interested event and then defines the objectives and constraints of the corresponding mathematical model [27].

We demonstrate the typical stochastic programming model by incorporating *uncertainty* in the form of linear programming. Various stochastic models are generated because of the specific demand of various industries. A set of classic and frequently used stochastic models with different constraints or objective functions will also be shown. Additionally, by this example (1.1), we are able to show you that if we replace a certain number b with a uncertain variable ξ, the complexity of the problem will grow exponentially. Furthermore only a few random variables may make the computational complexity of the model fall out of the capacity of the most powerful computer. The uncertainty could happen in A, c or b. That means, in reality, we could construct our model including random variables in any place of A, c or b. Since deterministic programming is prevalent, it may be tempting to suggest that random variables are replaced by their nominal statistics and hence solve the resulting problem with deterministic programming. Although this method might work in some situations, the solutions provided by this method are structurally different from those provided by stochastic programming models. To better understand the difference, please refer to an example provided in [30].

Example 1. *Consider a simple linear programming problem:*

$$\min \{2x_1 + 3x_2 | x_1 + x_2 = \xi\} \tag{1.3}$$

where ξ is a random variable which subjects to Poisson distribution.

In order to solve this model with uncertainty, we introduce another two variables, say y_1, y_2, into the model. y_1 means the shortage for the constraint $x_1 + x_2 = \xi$ and y_2 is surplus regarding to a constantly changing ξ. Because of the introduction of new variables in constraints, the objective function should make a change correspondingly. In mathematical terms, this is called a penalty. In other words, the penalty is the cost of the introduction of new variables. Here we define q^- as the unit cost of y_1 and q^+ as the unit cost of y_2. Then a new model is generated after the new variables and unit costs are introduced.

$$\min \{2x_1 + 3x_2 + q^- y_1 + q^+ y_2 | x_1 + x_2 + y_1 - y_2 = \xi\} \tag{1.4}$$

Suppose ξ is a set of random variables. The number of elements in the set is K. Each one in the set can be represented as $\xi_1, \xi_2, \cdots, \xi_K$ which follow a Poisson distribution. If we enumerate possible outcomes with meaningful probability $\xi_1, \xi_2, \cdots, \xi_k$, then the model can be rewritten as:

$$\begin{aligned}
&\min 2x_1 + 3x_2 + \sum_{i=1}^{K} q^- y_1^i P_i + \sum_{i=1}^{K} q^+ y_2^i p_i \\
&\text{subject to: } x_1 + x_2 + y_1^1 + y_2^1 = \xi^1 \\
&x_1 + x_2 + y_1^2 - y_2^2 = \xi^2 \\
&x_1 + x_2 + y_1^3 - y_2^3 = \xi^3 \\
&\cdots \\
&x_1 + x_2 + y_1^K - y_2^K = \xi^K
\end{aligned} \tag{1.5}$$

where $P_i(x = k) = \frac{e^{\lambda}\lambda^k}{k!}$ and λ is the arrival rate of ξ.

If we denote $Q(x_1, x_2, \xi^K) = \min\ q^+ y_1^k + q^- y_2^k$, then the model above can be reconstructed as:

$$\begin{aligned}
&\min 2x_1 + 3x_2 + \mathbb{E}[Q(x_1, x_2, \xi)] \\
&\text{subject to: } x_1 + x_2 + y_1^k - y_2^k = \xi^k, k = 1, \ldots, K
\end{aligned} \tag{1.6}$$

For convenience and notation simplicity, we denote that

$$G(x_1, x_2, \xi^i) = 2x_1 + 3x_2 + Q_i(x_1, x_2, \xi^i) \tag{1.7}$$

Then the final model can be rewritten as:

$$\begin{aligned} &\min \mathbb{E}[G(x,\xi)] \\ &\text{subject to:} Ax \leq c, \qquad (1.8) \\ &x \geq 0 \end{aligned}$$

where $G(x,\xi) : \mathbb{R}^n_+ \rightarrow \mathbb{R}$ is a convex function of x for any given $\xi \in (\Omega, \mathcal{F}, \mathbb{P})$ which is a probability space, $x \in \mathbb{R}^n_+$, $A \in \mathbb{R}^{m \times n}$, $c \in \mathbb{R}^m_+$.

The most obvious merit of stochastic programming is that uncertainty is constructed explicitly as part of the model and hence is considered in the process of decision making. Stochastic programming uses random variables as representatives of uncertainty. Since the basic premise of stochastic programing is that the probability of random variables is known, the random variables can be generated by their underlying probability. In two stage stochastic programming the decision maker takes some action in the first stage, after which a random event occurs affecting the outcome of the first stage decision. A recourse decision continues to be made in the second stage that compensates for any bad effects that might have been incurred as a result of the first-stage decision.

Model (1.8) is the general form of two stage stochastic linear programming problems, which have successful applications in portfolio management, risk management or revenue management[23]. For example, fleet management is one of the most critical and interesting problems in this area. The core of the fleet management problem includes managing fleets of equipment to satisfy customer requests as they evolve over time. The equipment represents a reusable resource, which serves customers (people or freight) who typically want to move from one location to the next. One of the random occurrences of fleet management is that customer requests arrive randomly. Sometimes orders arrive within a narrow interval, while other times there are no requests. In addition, the time of transportation is another random factor. We cannot predict what will happen during the movement of equipment, especially over long distance. In actual applications, there may be other sources of randomness such as equipment failures. So the high-dimensionality of the decisions involved has made fleet management a natural application for the techniques of stochastic programming. Although deterministic optimization models in transportation and logistics are often applied to the transportation and logistics problems, and the results are deemed "good enough" by decision-makers, stochastic optimization models can provide better solutions after introducing random variables.

However, the model optimizes an expected-value criterion and therefore the optimal solution is only an average. In selected applications, such as financial planning, decision makers often prefer modeling variance as a measurement of risk. Consider a situation: XYZ company needs to decide which stock to buy. The first stock has the same possibly of earning 1 dollar or losing 1 dollar per share with even chance. The second stock may make the company earn 1 million and 1 dollars or lose 1 million per share with the same possibility. The expectation of the first stock is \$0 and the expectation of the second is greater than \$0. According to the expected-value criterion, the company will choose the second stock. But in fact, it is quite dangerous for the company to invest on the second one at the risk of losing 1 million per share. This example tells us why it is necessary to investigate trade-offs between means and variances of costs (or profit). A model which integrates means and variance was introduced in [17] as a weighted mean-risk criterion.

The general model formation can be stated as follows:

$$\begin{aligned} &\min \mathbb{E}[G(x,\xi)] + \lambda \mathrm{Var}[G(x,\lambda)] \\ &\text{subject to: } Ax \leq c \qquad\qquad (1.9) \\ &x \geq 0 \end{aligned}$$

where Var(.) is a dispersion statistic and λ is a non-negative weight. It is typically intended as a measure of the decision maker's aversion to objective function variability. However there is an unpleasant consequence about the model (1.9) in that it is computationally intractable for even the simplest stochastic programs ([1]). In order to fix this problem, [2] defines the *coherent measure of risk* to preserve the convexity. In [2], it is presented the process of constructing a convexity preserved model, *coherent measure of risk.* Essentially, the purpose of replacing the variance by the *coherent* part is to preserve the convexity of (1.9) by defining a convex penalty function.

In other applications, it might be more appropriate to accept the possibility of infeasibility under some circumstances, provided the probability of this event is restricted below a given threshold. For example, in designing a call center, decision makers need to specify a time limitation. If the operation cannot pick up phones within the time limitation, the customer service level will be harmed. In this case, the mathematical model may be built with probabilistic constraints. The formation of this model can be constructed as an extension of a deterministic linear programming model. The general form of probabilistic constraints can be constructed as follows:

$$\min_{x \in X} c'x$$

$$\text{subject to: } P(Ax \le b) \ge p \tag{1.10}$$

The model under these constraints is quite problematic because the constraints are not convex. There are exceptions. When $P(Ax \le b)$ follows Normal distribution, then the problem is convex; therefore, it can be solved by various methods.

There is another model which is useful in various situations as well. For example, it is often applied in the portfolio selection with constraints on conditional value-at-risk (*CVaR*).

The formation is expressed as follows:

$$\min_{x \in X} \{f(x) | \mathbb{E}[G(x, \xi)] \leq b\} \tag{1.11}$$

Problems that can be solved but not fast enough for the solution to be useful are called intractable. In computer science, polynomial time refers to the running time of an algorithm, that is, the number of computation steps a computer or an abstract machine requires to evaluate the algorithm. Polynomial time is a synonym for "tractable", "feasible", "efficient", or "fast". Actually this model is generated based on the previous one. Since constraints in (1.10) make the problem computationally intractable, we use expectation instead of probability in order to reduce the computational size.

All the problem instances above may not be computationally tractable due to the exponentially growing problem scale. Let us consider the simplest instance, model (2.13). There are three elements in the set of ξ, namely $K = 3$. For each ξ, the arrival rate λ equals to 5, 10, 10 respectively. The problem scale is equal to 1.35×10^4 variables (or constraints for its dual). Since the computational complexity of the model with uncertainty expands exponentially, if we increase the size of the decision variables or ξ in this case, the complexity of the problem will easily reach the limitation of supercomputer.

To date, the fastest supercomputer is a system called "Roadrunner." This system is a Linux cluster which is capable of executing 12.8 G (1.28×10^{10}) Floating Point Operation Per Second (FLOPS). When solving programs with n variables, the computer is actually solving a $n \times n$ Newton system of linear equations. With standard techniques, the cost will be $O\{n^3\}$.

In mathematics and computer science, big O notation describes the limiting behavior of a function when the argument tends towards a particular value or infinity, usually in terms of simpler functions. Let f(x) and g(x) be two functions defined on some subset of the real numbers.

$$f(x) = O(g(x))x \rightarrow \infty \quad (1.12)$$

if and only if, for sufficiently large values of x, f(x) is at most a constant times g(x) in absolute value.

Certainly, when special structures, e.g. banded, and sparse, are identified, some factorization method can be applied to significantly reduce the computational overhead. Typical linear programming model in commercial scale always lead to a sparse Newton system and, as a result, the modern supercomputers are capable of solving commercial linear programming with 10^6 variables. Typical nonlinear convex programs, however, usually lead to a dense Newton system and the supercomputer "Roadrunner" is only capable of solving a nonlinear convex programming with 10^4 variables. Therefore, models with uncertainty would easily encounter the explosion of variables. We will introduce several prevailing solution techniques in section 1.3.

1.3. LITERATURE REVIEW

Uncertainty is an important element in decision-making problems. The data A and b (1.1) associated with a linear program are "uncertain" to some degree in most real world problems. In many models the uncertainty is ignored altogether. Ignoring uncertainty may lead to inferior or simply wrong decisions for some specific problems. There are a variety of ways in which the uncertainty can be formalized, and over the years various approaches to optimization under uncertainty were developed [28].

In this section, we are going to summarize the evolution of various techniques used to resolve problems with uncertainty. One easy way to solve problems with uncertainty is to replace uncertainty by nominal values (e.g., expected values) and solve the resulting deterministic model. However this method has been proven less effective than explicitly integrating uncertainty into the model like stochastic programming ([37], [27], [7] and references therein). Other classical approaches in operation research/management science for dealing with models with uncertainty are stochastic programming, robust optimization and probabilistic programming. Since robust optimization is based on totally different ideas from the other two, we do not consider this method in this thesis.

The first stochastic program with recourse was formulated in [11] and the model was named as linear programming under uncertainty, i.e. the two stage stochastic programming with recourse. The decision variables are partitioned into two sets. The first is that of variables that are decided prior to the realization of uncertain events. The second is the set of *recourse* variables which represent the optimal solution corresponding to the first stage decision and realized uncertainty. Other penalty methods are the *Scenario optimization* and the *Entropic Penalty methods* ([3], [26]). It is fair to say that stochastic programming solves a *relaxation* of the constraints ([18] and the references therein). All of the dominating penalty approaches cannot guarantee to recover original linear programming constraints but only to solve the relaxation of the constraints. The essential components of the typical stochastic programming model are ([37]):

(1) A decision vector that must satisfy certain constraints.

(2) A random variable ξ whose value will only be observed after the decision has been made.

(3) An evaluation (cost, possibly) of the decision in terms of the observed outcome.

The decision process involves a choice of a decision vector x, then ξ occurs and is observed. Finally a recourse action y is selected so as to satisfy the stochastic constraints. Since the decision y is completely determined by the selection of a given x and the occurrence of some ξ, the only real decision is the choice of x. The actual value of the decision is determined as soon as x and ξ are known, even though determining this value involves solving a linear program [32]. In mathematical terms, the general formulation of the second stage stochastic programming is expressed as follow [6]:

$$\begin{aligned}
&\min \{c'x + \mathbb{E}[\min q'y(\xi)]\} \\
\text{subject to: } & Ax \leq b, \\
& Tx + Wy(\xi) = h(\xi) \qquad (1.13) \\
& x \geq 0, y(\xi) \geq 0
\end{aligned}$$

where $c, T, q, h(\xi)$ are necessary coefficients with corresponding dimensions and $y(\xi)$ is called the *second stage decision* which is a function of x and ξ. Under proper assumption, $\mathbb{E}[\min q'y(\xi)]$ is convex of x for a given ξ. If we compare the solutions of stochastic programming with its deterministic counterpart, we can find that the solution from the stochastic program is $well - hedged$, building in some flexibility to meet the uncertain demand in the second stage. A second important observation from stochastic programming models is that the sequencing of decisions and observations is important. In constructing a stochastic programming model, it is not enough just to specify the decision variables. The modeller must also construct the model in such a way that prevents decisions that anticipate future uncertain events. A third point about the difference between stochastic and deterministic model is that the objective function does not account for the variation in outcomes. The model minimizes an expected cost, and its optimal policy is given under each scenario. And the solution might not be the best for some scenarios.

Two stage stochastic programming can also be extended into multi-stages. In the multi-stage stochastic programming, the uncertain parameters are revealed gradually over time. Then each stage (except the first stage) will contain some new uncertain parameters followed by the decision of previous stages.

As the applications of stochastic programming are widely spread, a new modelling approach using chance constraints is developed in parallel with the stochastic programming with recourse since the 1950s ([9], [10], [31], [35]). This approach addresses the problems of uncertainty by relaxing constraints with certain probability. It is also called probabilistic programming. The basic formulation of probabilistic programming is modeled as follows:

$$\min \{f(x)|\mathbb{P}(g_j(x,\xi) \leq 0) > 1-\alpha, j=1,\cdots,m\} \tag{1.14}$$

Here, x and ξ are decision and random vectors,respectively. α is a probability measure which typically ranges within [0, 0.10]. $g_j(x,\xi) \leq 0$ refers to a finite system of inequalities. These constraints do not require that the decisions are feasible for (almost) every outcome of the random parameters, but require feasibility with at least some specified probability. Many people find this method very appealing, but in most applications (although not all) it is questionable either from the modeling perspective or from the technical angle. In modeling, there is little guidance for what probability levels α should be, because no quantitative assessment is being made of the consequence of having $g_j(x,\xi) \leq 0$.

There are technical difficulties of dealing with *chance-constraints* as well. Not surprisingly, a general solution method for chance constrained programming does not exist. There are multiple reasons that the probabilistic programming is difficult to solve. First, it becomes too costly to evaluate whether or not a given constraint is satisfied at a given point x. Second, the feasible set of constraints cannot always preserve the convexity even when $g_j(x,\xi)$ is affine. Therefore, the model under this constraint is highly problematic. When the distribution of ξ is *logarithmically concave*, however, the feasible set of a chance constraint of an affine function is convex ([20], [24]) and the model (1.14) can be rewritten into a quadratic conic programming and solved by interior point methods.

Different from other stochastic programs, those with chance constraints need two kinds of approximations: first, the distribution of the random parameter is almost never known exactly and has to be estimated from a large amount of historical data. Secondly, even a given multivariate distribution (such as multivariate normal) cannot be calculated exactly in general but has to be approximated by simulations or bounding arguments. Both types of imprecision motivate the discussion of stability in programs with chance constraints. There are numerous applications of chance constrained programming such as energy production, finance and transportation([25]).

The choice of solution methods strongly depends on how random and decision variables interact in the constraint model. The problem scale of two stage stochastic programs (1.8) is quite large, and expectation function can not be calculated exactly for most cases. If the number of possible scenarios of ξ an number of variables are not large enough, (1.8) can be rewritten into an equivalent linear programming model ([18],[6]). Obviously, this method is not tractable for realistic problems with large numbers of scenarios.

Since this class of problems are typically very large in scale, many researches have been focused on developing algorithms that exploit the problem structure, hoping to decompose large problems into smaller more tractable components. Due to large number of scenarios, proper sampling based methods are employed. Specifically, the *external sampling method* and the *internal sampling method* are employed. The external sampling method estimates the objective function by Monte Carlo simulation and the internal sampling method estimates the gradient/subgradient of the objective. The external sampling method typically takes one sample before applying a mathematical programming method and the internal sampling method updates the estimation from iteration to iteration. Many algorithms have been developed to take repeated samples during the course of the algorithm. Details of the convergence properties of the external and internal sampling methods can be found in [19] and [32]. A method such as sample average approximation belongs to the class of external sampling method, while stochastic approximation and stochastic decomposition are both in the class of internal sampling method.

Our approximation method is an external sampling method because we use Monte Carlo sampling to estimate the objective and thereby apply the polynomial approximation on the simulation data. The resulting model, however, is different from the traditional external sampling approaches. Our model will be a convex optimization problem with polynomial constraints/objective which is highly tractable for most commercial solvers. Furthermore, our polynomial approximation can achieve arbitrary accuracy *uniformly*. The procedure is fundamentally based on the properties of Bernstein Polynomial, such as *shape-preserving* ([21]).

1.4. OBJECTIVE

The goal of the thesis is to present a novel combination of methods to solve two stage stochastic programs in large scale. First of all, Monte Carlo based sample average algorithm is applied to simulate the objective function. Secondly, Bernstein polynomial is used to approximate the simulation data. In the end, gradient descent method is applied to solve the resulting Bernstein polynomial approximation.

The most important part of the proposed methodology is to obtain convex polynomial approximations with arbitrary accuracy *uniformally*. Prior to introducing Bernstein polynomial approximation, we will make a review of all available sampling method such as *SAA*, *SA* and *SD*. Numerical results of *SAA* in different scenarios will be presented in chapter 3. After we get Bernstein polynomial approximation, gradient descent method is implemented in order to solve the tractable convex function. The complete theoretical proof of this algorithm is available in chapter 2.

In order to obtain an arbitrary approximation, we need to determine the necessary degree for each approximation polynomial. By Weierstrass theorem and our finding, the polynomial with higher degree will yield better approximation to to the original function. However, it may incur bad effects as well such as consuming more time and large model scale. The result is promising in both theory and practice. We employ a specific example to demonstrate how to calculate the reasonably small necessary degree in chapter 3.

Lastly, we need to show the advantage of our approach against the prevailing Sample Average Approximation (SAA) method. We show that the our approximation procedure will deliver outstanding numerical performance and converge to optimal solution reasonably fast.

CHAPTER 2

MODEL

In this part we will carefully consider stochastic programming optimization problem in this form:

$$\min_{x \in X} \{f\{x\} := \mathbb{E}[G(x,\xi)]\} \tag{2.1}$$

Here X is a finite subset of $\mathbb{R}^n$ such that $X = \{x | g_j(x) \leq 0, j = 1, \cdots, m\}$, $G(x,\xi)$is a real valued measurable function of two (vector) variables x and ξ. ξ is a random vector in $\mathbb{R}^s$ whose distribution is known denoted as P. $\mathbb{E}[G(x,\xi)] = \int G(x,\xi) dP(\xi)$ is the corresponding expected value. It follows that the function $f(\cdot)$ is convex and has finite valued on X. It is typically impossible to calculate the expected value $\mathbb{E}f(x,\xi)$ in a closed form [34].

There are some basic properties of the model (2.1). First, the objective function $\mathbb{E}[G(x,\xi)]$ is convex. Second, when the distribution of ξ is discrete and $G(x,\xi)$ is piecewise convex linear, then $\mathbb{E}[G(x,\xi)]$ is a piecewise linear function as well. For piecewise linear functions, we can use its equivalent linear programming model presented in both [5] and [14] as a substitute. If the number of variables is limited and the number of constraints are huge, we can apply the cutting plane method to solve it.

There are other methods for it as well. *SAA* is an approach for solving stochastic problems by using Monte Carlo simulation. The *SAA* method sample certain amount of *representing* scenarios and solves the reduced problems. The asymptotic optimality and convergence are proved in [33]. In this technique the expected function of the stochastic problem is approximated by a sample average estimate derived from a random sample. The resulting sample average approximating problem is then solved by other optimization approaches. The process is repeated with different samples to obtain candidate solutions along with statistical estimates of their optimality gaps until a stopping criterion is satisfied.

SAA is a valued method and the idea of using *SAA* for solving stochastic programming problems is applied by various researchers to numerous stochastic problem ([15]). From theoretical studies and numerical experiment results, *SAA* method prove itself reasonably efficient for solving certain classes of two stage stochastic problems integrated with a good deterministic algorithm ([12]). Moreover, *SAA* is also applied to solve model expressed as follows:

$$\min \{f(x)|\mathbb{E}[G(x,\xi)] \leq b\} \tag{2.2}$$

which is a typical mathematical model raised from the portfolio selection with constraints on conditional value-at-risk ([36]). Another class of problems that *SAA* works well is the probabilistic problem or chance constrained problem ([16]).

The procedures of applying *SAA* are fairly simple. Suppose we generate an independent identically distributed random (*iid*) sample $\xi^1, \cdots, \xi^N$ of N realizations of the considered random vector; hence, we can estimate the expected value function $f(x)$ by the sample average as follows:

$$\hat{f}_N(x) := \frac{1}{N}\sum_{j=1}^{N} F(x,\xi^j) \tag{2.3}$$

Consequently, we change the original problem into a deterministic problem this way:

$$\min_{x\in X} \hat{f}_N(x). \tag{2.4}$$

However, it should be noticed that *SAA* is not an algorithm, the obtained *SAA* problem still has to be solved by an appropriate numerical procedure.

In addition to SAA method, there are other methods such as *SA* and *SD* method developed to solve stochastic programming. Different from SAA method as an external sampling method, these two methods are internal sampling methods. Comparing with external sampling methods, internal sampling methods apply sampling during each iteration to estimate the gradient and bounds. Practically, *SD* method's performance is not stable due to possible bad estimations on gradients. And *SA* is a crude sub-gradient method which often performs poorly in practice.

SD method decomposes the stochastic elements of a problem into the deterministic data in a manner that is reminiscent of the separation of the integer variable from the continuous variables in mixed integer programming under Benders' decomposition ([29]). However, unlike Bender's decomposition, *SD* combines successive approximation methods of mathematical programming with sampling approaches commonly adopted in the statistical literature. The manner in which *SD* uses to sample data makes this method extremely flexible in its ability to accommodate various representations of uncertainty. *SA* is the stochastic version of gradient type method which is crude in practice.

All of these methods have their own problems respectively, particularly when the problem scale grows exponentially. *SAA* method actually reduces the original problem down to make the model computationally tractable. The optimal solution is a random variable and its interval estimation is an on-going research topic. *SA* and *SD* methods require evaluating the subgradient through sampling. In practice, the estimated subgradient may actually penetrate the supporting hyperplane. In addition, *SA* method appears more likely to be slow in convergence. If the model is simple recourse, it becomes computationally efficient([7], [13]); however, such simple recourse model rarely appears in business decisions.

This thesis applied a polynomial approximation approach to solve two stage stochastic programming model. The demand of approximating the objective by polynomial is from the tractability issue. We need to evaluate the gradient/sub-gradient or even Hessian accurately and efficiently in order to solve the convex programming, no matter with differentiable or non-differentiable constraints. Due to the numerous realizations, sampling methods are widely applied in estimating the gradient/sub-gradient. On the other hand, the polynomial is a *transparent* function and its derivatives can be easily evaluated. When for any ξ and a large n,

$$|f(x) - \mathbf{P}_n(x)| \leq \xi \tag{2.5}$$

where $\mathbf{P}_n(x) = \sum_{i=1}^{n} a_i x^i$ and n is called the degree if $a_n \neq 0$. The numerical properties of $\mathbb{P}_n(x)$ are expected to be consistent with $f(x)$. The idea is not new and there have been several approaches being developed, such as the least square, least norm approximation, and the interpolation. There is another polynomial approximation method which are based on Bernstein polynomials of the objective function. It has been proven in [21] that using Bernstein polynomials can generate a better approximation than least square method. The main difference between these approaches exists in the way of choosing non-negative multiples and component functions. However, these methods except Bernstein polynomial approximation are not useful for convex programs because of their poor performance on approximating the derivatives. Although the least square method can provide us a polynomial approximation function, the result is not satisfactory. Least square method can only guarantee the approximation of each point is controlled within the deviation of ε from the original value. The estimation on gradient/subgradient is poor and increasing the degree of polynomial will worsen the problem.

For convex problems, a qualified polynomial approximation should be able to

(1) Preserve the convexity of the objective and constraints.

(2) Arbitrary accuracy for the function,its gradient/sub-gradient and Hessian (if twice differentiable)

(3) Uniform convergence for the sequence of polynomials by degrees on a closed interval.

Except the requirements listed above, the arbitrary accuracy polynomial should be constructed within a reasonably finite degree at a given $\varepsilon > 0$, denoted by $M(\varepsilon)$.

In some applications, $\mathbb{E}[G(x,\xi)]$ might be estimated by Monte Carlo, like $\frac{1}{N}\sum_{i=1}^{N} g(x,\xi^i)$ where N is the number of the replications. In this case, we need to show that our approximation is still valid. What's more, the number of replications N and the necessary degree $M(\varepsilon)$ should be determined at a given ε. Afterwards, we can transform the objective function into a polynomial approximation function and apply convex optimization techniques.

Consider a convex function $f(x) : [a,b] \to \mathbb{R}$ on a closed interval $[a,b]$, we can make an affine change of variable to transform $[a,b]$ onto $[0,1]$. Then we have the following definitions and theorems:

Definition 1. *The Bernstein polynomial of degree n are defined by*

$$B_{i,n} = \begin{pmatrix} n \\ j \end{pmatrix} t^j (1-t)^{n-j} \tag{2.6}$$

for $i = 0, 1, \cdots, n$, where

$$\begin{pmatrix} n \\ j \end{pmatrix} = \frac{n!}{j!(n-j)!}$$

It is also known that if $f(x)$ is convex, so is its Bernstein polynomial,

$$B_n(f;x) = \sum_{j=0}^{n} \begin{pmatrix} x \\ y \end{pmatrix} x^j (1-x)^{n-j} f(j/n), \tag{2.7}$$

for any $n = 0, 1, 2, \cdots$.

Lemma 1. *Given a continuous function $f(x) : [0,1] \to \mathbb{R}$ and any $\varepsilon > 0$, there exists an integer $M(\varepsilon) = \dfrac{\sup|f(x)|}{\varepsilon\delta^2}$ such that when $n \geq M(\varepsilon)$*

$$|f(x) - B_n(f;x)| \leq \varepsilon,$$

$\forall x \in [0,1]$ *where* $\delta(\varepsilon, f(\cdot)) > 0$ *is determined by the uniform continuity.*

Note that n is the degree of the Bernstein polynomials which must satisfy the following inequality.

$$n \geq M(\varepsilon) \tag{2.8}$$

where $M(\varepsilon) = \dfrac{\sup|f(x)|}{\varepsilon\delta^2}$ and δ is determined by the property of *uniform continuous* (see [22]). We will show the calculation of $M(\varepsilon)$ in Chapter 3 for an example in Logistics.

Lemma 2. *If* $f(x)$ *is twice continuously differentiable, then* $B_n'(f;x), B''(f;x)$ *converge uniformly to* $f'(x), f''(x)$ *respectively.*

According to [21], the basic idea of Bernstein polynomial approximation method can be summarized as follows: Suppose $f(x)$ is the objective function convex on a finite interval $[a,b]$ that needs to be approximated. And $\psi_0(x), \psi_1(x), \psi_2(x), \ldots,$ are a sequence of functions which should be convex on $[0,1]$. We may use functions in this form:

$$\phi_n(x) = \sum_{j=0}^{n} c_j \psi_j(x), c_j \geq 0 \tag{2.9}$$

to approximate $f(x)$. $\phi(x)$ still preserves convexity since we just sum up non-negative multiples of the component functions $\psi_j(x)$.

Theorem 1. *There exists a sequence of component functions,*

$$\psi_0(x), \psi_1(x), \psi_2(x), \cdots,$$

each convex on $[0,1]$*, such that any function* $f(x)$ *which is convex on* $[0,1]$ *may be approximated with arbitrary accuracy on* $[0,1]$ *by the sum of nonnegative multiples of the component functions.*

The proof is provided in [21] and [22]. Based on it, we develop our procedure to construct the polynomial approximation.

Procedure 1. *Since the decision variable $x = (x_1, \ldots, x_n)'$, we need to construct a non-negative linear combination of convex polynomials for every component $x_i, i = 1, \ldots, n$. First, we choose $\ell + 1$ distinct $x_i^0, x_i^1, \ldots, x_i^\ell \in [a_i, b_i], x_i^0 = a_i, x_i^\ell = b_i, n \geq M(\varepsilon/n)$.*

$$\psi_j(x_i^s) = \left[\frac{x_i^s - a_i}{b_i - a_i}\right]^j \sum_{k=0}^{n-j} \frac{(-1)^k \left[\frac{x_i^s - a_i}{b_i - a_i}\right]^k \binom{n-j}{k}}{(k+j)(k+j-1)}, 2 \leq j \leq n \tag{2.10}$$

$$\psi_1(x_i^s) = x_i^s sign[\mathbb{E}(g_i(a_i, \xi))], \psi_0(x_i^s) = sign[\mathbb{E}(g(a_i, \xi))], s = 0, \ldots, \ell \tag{2.11}$$

Then, we approximate $\mathbb{E}[g_i(x_i, \xi)]$ (the i^{th} component of $\sum_{i=1}^n \mathbb{E}(g_i(x_i, \xi))$) on $[a_i, b_i]$ by (2.12). *For any $\varepsilon > 0$,*

$$P_{M_i(\varepsilon/n)}(x_i) = \sum_{j=0}^{M_i(\varepsilon/n)} c_i^j \psi_j(x_i) \tag{2.12}$$

where $c_i^j, i = 1, \ldots, n,\ j = 0, \ldots, M_i(\varepsilon/n)$ are the corresponding nonnegative coefficients for the polynomial. c_i^j is an optimal solution from the norm approximation model (2.13).

$$\min \sum_{s=0}^{\ell} \left\{ \sum_{i=0}^{M_i(\varepsilon/n)} c_i^j \psi_j(x_i) - \mathbb{E}[g_i(x_i, \xi)] \right\}^2$$

$$\text{subject to: } c_i^2, \ldots, c_i^{M_i(\varepsilon/n)} \geq 0 \tag{2.13}$$

where we choose $\forall \varepsilon > 0, M_i(\varepsilon/n)$ which is large enough to approximate $\mathbb{E}[g_i(x_i, \xi)]$ within ε/n degree of Bernstein polynomial. Model (2.13) *is a norm approximation problem in [8] which can be solved by multiple polynomial algorithms. In realistic problems, $x_i^0, x_i^1, \ldots, x_i^\ell \in [a_i, b_i]$ are the points we have special interest and they are regarded as perturbed values of some differentiable convex function. We must select these points with good reasons. For example, when integer constraints exist, we will study the integer points and their neighborhoods. Therefore, $x_i^0, x_i^1, \ldots, x_i^\ell \in [a_i, b_i]$ are likely to be corresponding integer points and our approximation should be calculated accordingly afterwards.*

For most realistic problems, evaluating the objective $\sum_{i=1}^{n} \mathbb{E}[g_i(x_i,\xi)]$ is difficult. For example, when $g(x,\xi) = \min\{x,\xi\}$, this function can not be precisely evaluated. We obtain the function value component wise by Monte Carlo simulation with N_i for the i^{th} component which can be generally expressed in this way:

$$\mathbb{E}[g_i(x_i,\xi)] \approx \frac{1}{N_i}\sum_{i=1}^{N_i} g_i(x_i,\xi^{N_i}) \text{ when } N_i \text{ is large enough.} \tag{2.14}$$

There is an unpleasant consequence that the simulated data may lose convexity. The remedy is to simulate the function with large enough number of replications and thereby apply the previously introduced procedure, we can still have nonnegative linear combination of convex polynomials.

After applying Bernstein polynomial approximation, the original model is transformed into

$$\begin{aligned} &\min P_n(x) \\ &\text{subject to: } Ax \leq b \\ &x \geq 0 \end{aligned} \tag{2.15}$$

where $P_n(x)$ is the resulting polynomial approximation. Since its gradient/subgradient can be evaluated, we will apply the first order gradient method on it.

Suppose that we are at the point $x = \bar{x}$, where $A\bar{x} = b$, i.e., $\bar{x}$ is a feasible point. Then we have

$$f(\bar{x}+d) \approx f(\bar{x}) + \nabla f(\bar{x})^T d \tag{2.16}$$

for d "small". In order to choose the direction $\overline{d}$ and compute the next point

$$x' = \bar{x} + \alpha\overline{d} \tag{2.17}$$

for some stepsize α, we will solve the following direction-finding problem:

$$\begin{aligned} &\min f(\bar{x}) + \nabla f(\bar{x})^T(x-\bar{x}) \\ &\text{subject to: } Ax = b \\ &\qquad\qquad \|x-\bar{x}\| \leq 1, \end{aligned} \tag{2.18}$$

Note that $Ad = 0$ ensures that $A(\bar{x}+\alpha d) = A\bar{x} = b$ for any α. Also note that the constraint "$d^T Id \leq 1$" says that d must lie in the Euclidean unit ball B, define as:

$$B = \{d \in \Re^n | d^T Qd \leq 1\}, \tag{2.19}$$

where Q is a given symmetric and *positive-definite* matrix. This lead to the more general direction-finding problem:

$$\begin{aligned} &\min \nabla f(\bar{x})^T d \\ &\text{subject to: } Ad = 0 \\ &\qquad\qquad d^T Qd \leq 1. \end{aligned} \tag{2.20}$$

The **projected steepest descent algorithm** is:

(1) *Step 1.* $\bar{x}$ satisfies $A\bar{x} = b$. Compute $\nabla f(\bar{x})$.

(2) *Step 2.* Solve the direction-finding problem (DFP):

$$\bar{d} = \arg\operatorname{minimum} \nabla f(\bar{x})^T d$$
$$\text{subject to: } Ad = 0$$
$$d^T Q d \leq 1, \tag{2.21}$$

If $\nabla f(\bar{x})^T \bar{d} = 0$, stop. In this case, $\bar{x}$ is a Karush-Kuhn-Tucher point.

(3) *Step 3.* Solve $\min_a f(\bar{x} + \alpha\bar{d})$ for the stepsize α, perhaps chosen by an exact or inexact linesearch.

(4) *Step 4.* Set $\bar{x} \leftarrow \bar{x} + \alpha\bar{d}$. Go to *Step 1.*

By steepest descent method, we can get the optimal value and solutions of the resulting Bernstein polynomial approximation; therefore, the original two stage stochastic programming can be solved completely. A realistic logistics problem will be solved by the proposed methodology and numerical results will be presented in chapter 3.

Comparing with gradient descent method as the first order gradient method, Newton method as the second order gradient method can be more effective to find a "better" direction in each step size; therefore, reach the optimal point more quickly. It is tempting to apply Newton method to the resulting Bernstein polynomials. In order to apply the procedure, the original function needs to be differentiable. Consider the significant proportion of two stage stochastic programming's objectives are piecewise linear and non-differentiable, although the estimation on subgradient is correct, the estimation on Hessian will be unstable and unbounded. Therefore, we can apply the second order algorithms by applying our procedure only if the original function is continuously differentiable.

CHAPTER 3

NUMERICAL STUDY

In this chapter, we will present a numerical study using the proposed methodology for solving a real-world problem. We have the following problem instance,

$$\begin{aligned} \min \; -\sum_{i=1}^{60} \mathbb{E}[f_i \min(x_i, \xi_i)] & \\ \text{subject to: } Ax \le c & \\ x \ge 0, & \qquad (3.1) \end{aligned}$$

where ξ is a 60-dimensional Poisson random vector. A is a 10×60 matrix. $c = (400, \cdots, 400)'$ is a 10×1 vector and f is a 60×1 nonnegative vector. We summarize the matrix A and the other settings on c, f, ξ in table 3.1,

The problem (3.1) can be described as the following applications.

Example 2. Network Revenue Management: *Suppose we have 10 resources that means flight legs in the airline application and 60 products. A production consists of a seat on one or several flight legs in combination with a fare class and departure date. Each resource has the same capacity 400, and the network capacity is given by the corresponding vector* $c = (400, \cdots, 400)'$ *which is a* 10×1 *vector. Every product has an associated revenue f. By defining* $a_{ij} = 1$ *if resource i is used by product j, and* $a_{ij} = 0$ *otherwise, we obtain the incidence matrix* $A = (a_{ij}) \in 0, 1^{m \times n}$. *We assumes that each product uses at most one unit of any resource, so* a_{ij} *is either 0 or 1.* $\mathbb{E}[\xi]$ *means the arrival rate of a customer in any time period.*

Variables	R1	R2	R3	R4	R5	R6	R7	R8	R9	R10	$\mathbb{E}(\xi)(units)$
$x_1(\$300), x_{31}(\$800)$	√										40(1:3)
$x_2(\$300), x_{32}(\$800)$		√									40(1:3)
$x_3(\$300), x_{33}(\$800)$			√								40(1:3)
$x_4(\$300), x_{34}(\$800)$				√							40(1:3)
$x_5(\$300), x_{35}(\$800)$					√						40(1:3)
$x_6(\$300), x_{36}(\$800)$						√					40(1:3)
$x_7(\$300), x_{37}(\$800)$							√				40(1:3)
$x_8(\$300), x_{38}(\$800)$								√			40(1:3)
$x_9(\$300), x_{39}(\$800)$									√		40(1:3)
$x_{10}(\$300), x_{40}(\$800)$										√	40(1:3)
$x_{11}(\$500), x_{41}(\$100)$	√			√							100(1:3)
$x_{12}(\$500), x_{42}(\$100)$	√				√						100(1:3)
$x_{13}(\$500), x_{43}(\$100)$	√					√					100(1:3)
$x_{14}(\$500), x_{44}(\$100)$	√						√				100(1:3)
$x_{15}(\$500), x_{45}(\$100)$		√	√								100(1:3)
$x_{16}(\$500), x_{46}(\$100)$			√			√					100(1:3)
$x_{17}(\$500), x_{47}(\$100)$			√					√			100(1:3)
$x_{18}(\$500), x_{48}(\$100)$			√							√	100(1:3)
$x_{19}(\$500), x_{49}(\$100)$		√			√						100(1:3)
$x_{20}(\$500), x_{50}(\$100)$				√	√						100(1:3)
$x_{21}(\$500), x_{51}(\$100)$					√			√			100(1:3)
$x_{22}(\$500), x_{52}(\$100)$					√					√	100(1:3)
$x_{23}(\$500), x_{53}(\$100)$		√					√				100(1:3)
$x_{24}(\$500), x_{54}(\$100)$				√			√				100(1:3)
$x_{25}(\$500), x_{55}(\$100)$						√	√				100(1:3)
$x_{26}(\$500), x_{56}(\$100)$							√			√	100(1:3)
$x_{27}(\$500), x_{57}(\$100)$		√							√		100(1:3)
$x_{28}(\$500), x_{58}(\$100)$				√					√		100(1:3)
$x_{29}(\$500), x_{50}(\$100)$						√			√		100(1:3)
$x_{30}(\$500), x_{60}(\$100)$								√	√		100(1:3)
Availability (units)	400	400	400	400	400	400	400	400	400	400	

Table 3.1. Network Structure and Constraints for the logistic example

Example 3. Network Resource Allocation: *Various applications are raised from network resource allocation, such as transportation industry. Suppose there are 10 locations in the network, the* 10×1 *vector represents the upper boundary of available resource of each location. The availability of the path connected with each location is represented by* $A = (a_{ij}) \in 0, 1^{m\times n}$*, and* a_{ij} *is either 0 or 1. There are 60 customers who will place orders at the arrival rate of* $\mathbb{E}[\xi]$. f *is the unit cost for each customer.*

Example 4. Production Control: *Two stage stochastic model are also often employed in production control. For instance, there are 60 products which are manufacted through certain procedures. All of the available procedures are treated as resources; hence, there are 10 resources and the upper boundary of each resource is 400. We denote resources by* $c = (400, \cdots, 400)'$, *and* $A = (a_{ij}) \in 0, 1^{m \times n}$ *represents the necessary procedures for each product.* $\mathbb{E}[\xi]$ *means the arrival rate of the yield.* f *is the unit cost for each product.*

First, we apply *SAA* method to this problem. It is solved 5 times with different sample sizes. The performance of each scenario is listed in the table 3.2.

Sample Size	Optimal Value	Standard Deviation of Results
10	4.1723e+05	308.6
50	4.1582e+05	129.3
100	4.1539e+05	101.1
150	4.1542e+05	75.0
200	4.1530e+05	68.9

Table 3.2. Results of optimal value and standard deviation by *SAA* approach

Now we demonstrate the numerical result of the proposed method. First, determining the necessary degree for every approximation polynomial is a critical step. We show numerical result in determining the necessary degree and the theoretical analysis is in Chapter 2. For the component $\mathbb{E}[min\{x_1, \xi_1\}]$ where the arrival rate is $\lambda_1 = 15$, we are interested in the closed interval $x \in [13, 18]$. Within the interval, we choose $\varepsilon = 0.1$, then for any two distinct points a, b such that $|a - b| < \delta = 0.3$, we have $|\mathbb{E}[min\{a, \xi_1\} - \mathbb{E}[min\{b, \xi_1\}]| \leq = 0.1$. Then, $\sup|\mathbb{E}[min\{x_i, \xi\}]| \leq 0.1$. The necessary degree should be $\frac{0.1}{\delta^2\varepsilon} \leq \frac{0.1}{0.1 \times 0.3^2} = 11.11$ which means the degree should be at least 12. We present the graphical illustration of the polynomial approximations with degree of $n = 5, 10, 15$ respectively.

Figure 3.1, 3.2 and 3.3 show the Bernstein polynomial approximation curves with comparison of simulated data. The solid curves are drew from simulated data and the dotted curves are drew from our polynomial approximation procedure (see Procedure 1. From

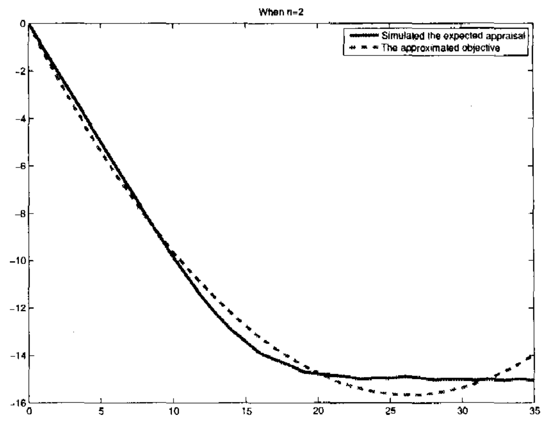

Figure 3.1. Polynomial Approximation at degree of 5

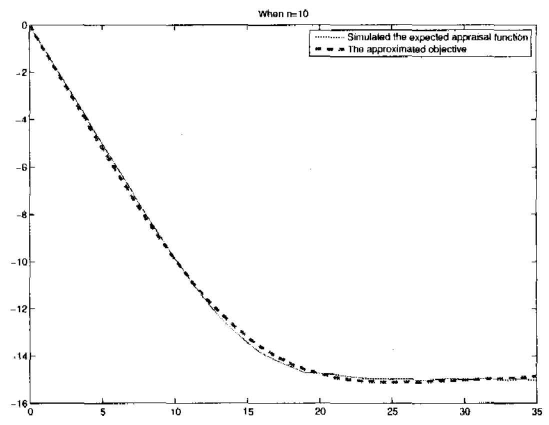

Figure 3.2. Polynomial Approximation at degree of 10

these figures, Bernstein polynomial approximation performs *uniformly* well when degree is 15. Numerically, the result is better than we expected. This is the uniform conclusion that when $n \geq 12$, the polynomial approximation will *uniformally* converge to the corresponding convex function with $\varepsilon = 0.1$.

After we got the simulation data, we will apply Bernstein approximation method and gradient descent method thereafter. We summarize the results in Table 3.3 by enumerating $n = 1, \ldots, 15, \varepsilon = 0.1, 0.01, 0.001$.

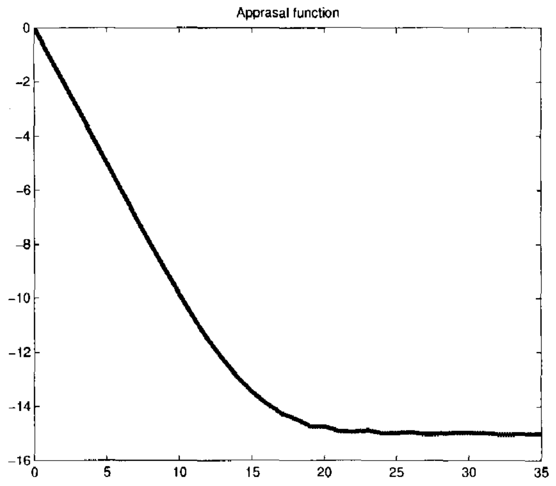

Figure 3.3. Polynomial Approximation at degree of 15

	$\varepsilon = 0.1$	$\varepsilon = 0.01$	$\varepsilon = 0.001$
	Optimal Value	Optimal Value	Optimal Value
$n = 1$	3.6955e+05	3.7033e+05	3.7015e+05
$n = 2$	4.1644e+05	4.1621e+05	4.1560e+057
$n = 3$	4.1676e+05	4.1664e+05	4.1606e+05
$n = 4$	4.1576e+05	4.1600e+05	4.1593e+05
$n = 5$	4.1518e+05	4.1588e+05	4.1694e+05
$n = 6$	4.1606e+05	4.1549e+05	4.1605e+05
$n = 7$	4.1574e+05	4.1622e+05	4.1655e+05
$n = 8$	4.1591e+05	4.1590e+05	4.1574e+05
$n = 9$	4.1497e+05	4.1550e+05	4.1556e+05
$n = 10$	4.1593e+05	4.1579e+05	4.1559e+05
$n = 11$	4.1553e+05	4.1576e+05	4.1516e+05
$n = 12$	4.1559e+05	4.1584e+05	4.1559e+05
$n = 13$	4.1584e+05	4.1529e+05	4.1540e+05
$n = 14$	4.1519e+05	4.1517e+05	4.1575e+05
$n = 15$	4.1357e+05	4.1357e+05	4.1498e+05

Table 3.3. The results of optimal value by proposed methodology

The optimal value of the model is \$ 415410. When we approximate the convex objective by a polynomial with degree of 12, the optimal solution (3.1) is \$ 415590 and the error is $\frac{415590}{415410} - 1 = 0.0004$ or 0.04 %.

Until now, we have demonstrated our proposed methodology's efficiency when encountering non-differentiable (for example, piece-wise linear) convex objective. We conclude that our approach would be a tractable alternative to some well established methods, such as SAA, SA and SD.

REFERENCES

[1] S. AHMED, *Convexity and decomposition of mean-risk stochastic programs*, Math. Program., 106 (2006), pp. 433–446.

[2] P. ARTZNER, E. F. DELHAEN, J-M, AND D. HEALTH, *Coherent measure of risk*, Mathematical Finance, 9 (1999), pp. 203–228.

[3] A. BEN-TAL, *The entropic penalty approach to stochastic programming*, Math. Oper. Res., 10 (1985), pp. 263–279.

[4] A. BEN-TAL AND A. NEMIROVSKI, *Robust solutions of linear programming problems contaminated with uncertain data*, Mathematical Programming, 88 (2000), pp. 411–424.

[5] D. BERTSIMAS AND J. N.TSITSIKLIS, *Introduction to Linear Optimization*, Athena Scientific, 1997.

[6] J. BIRGE AND F. LOUVEAUX, *Introduction to Stochastic Programming*, Springer-Verlag, Berlin, 1997.

[7] J. R. BIRGE AND F. LOUVEAUX, *Introduction to stochastic programming*, Springer-Verlag, New York, 1997.

[8] S. BOYD AND L. VANDENBERGHE, *Convex Programming*, Cambridge University Press, 2004.

[9] A. CHARNES, W. COOPER, AND G. SYMONDS, *Cost horizons and certainty equivalents: An approach to stochastic programming of heating oil*, Management Science, 4 (1958), pp. 183–195.

[10] A. CHARNES AND M. KIRBY, *Optimal decision rules for the triangular E-model of*

chance-constrained programming, Cah. Cent. Etud. Rech. Oper. 11, 215-243, (1969).

[11] G. DANTZIG, *Linear programming under uncertainty*, Management Science, 1 (1955), pp. 197–206.

[12] B. DELYON AND A. JUDITSKY, *Stochastic optimization with averaging of trajectories*, Stochastics Stochastics Rep., 39 (1992), pp. 107–118.

[13] P. KALL AND S. WALLACE, *Stochastic Programming*, Wiley, Chichester etc., 1994.

[14] L. KALLENBERG, *Survey of linear programming for standard and nonstandard Markovian control problems. I. Theory*, Z. Oper. Res., 40 (1994), pp. 1–42.

[15] A. J. KLEYWEGT, A. SHAPIRO, AND T. HOMEM-DE MELLO, *The sample average approximation method for stochastic discrete optimization*, SIAM J. Optim., 12 (2001/02), pp. 479–502 (electronic).

[16] J. LUEDTKE, S. AHMED, AND G. NEMHAUSER, *An integer programming approach for linear programs with probabilistic constraints.* Optimization Online, `http://www.optimization-online.org`, 2007.

[17] H. MARKOWITZ, *Portfolio selection*, Journal of Finance, 7 (1952), pp. 77–91.

[18] K. MARTI AND P. KALL, eds., *Stochastic programming*, vol. 423 of Lecture Notes in Economics and Mathematical Systems, Berlin, 1995, Springer-Verlag. Numerical techniques and engineering applications.

[19] D. MORTON AND E. POPOVA, *Monte carlo simulations for stochastic optimization*, in Encyclopedia of Optimization, C. Floudas and P. Pardalos, eds., Kluwer Academic Publishers, 2001.

[20] A. NEMIROVSKI AND A. SHAPIRO, *Convex approximations of chance constrained programs*, SIAM J. Optim., 17 (2006), pp. 969–996 (electronic).

[21] G. PHILLIPS AND P. TAYLOR, *Approximation of convex data*, BIT, 10 (1970), pp. 324–332.

[22] G. M. PHILLIPS, *Interpolation and Approximation by Polynomials*, Springer, 2003.

[23] W. B. POWELL AND H. TOPALOGLU, *Fleet management*, in Applications of stochastic programming, vol. 5 of MPS/SIAM Ser. Optim., SIAM, Philadelphia, PA, 2005, pp. 185–215.

[24] A. PREKOPA, *On logarithmic concave measures and functions.*, Acta Sci. math. 34, 335-343, (1973).

[25] A. PREKOPA, *Optimization under probabilistic constraints and its applications in statistics.*, in Computational statistics, Proc. 6th Symp., Prague 1984, 446-457, 1984.

[26] R. ROCKAFELLAR AND R. J.-B. WETS, *Scenarios and policy aggregation in optimization under uncertainty*, Math. Oper. Res., 16 (1991), pp. 119–147.

[27] A. RUSZCZYNSKI AND A. SHAPIRO, eds., *Stochastic Programming*, vol. 10 of Handbooks in Operations Research and Management Science, Elsevier, 2003.

[28] A. RUSZCZYŃSKI AND A. SHAPRIO, *Stochastic Programming, Handbooks in Operations Research and Management Science Volume 10*, Elsevier, New York, NY, 2003.

[29] J. H. S. SEN, *Stochastic decomposition*, vol. 8 of Nonconvex Optimization and its Applications, Kluwer Academic Publishers, Dordrecht, 1996. A statistical method for large scale stochastic linear programming.

[30] S. SEN AND J. L. HIGLE, *An Introductory Tutorial on Stochastic Linear Programming Models*, Interfaces, 29 (1999), pp. 33–61.

[31] J. SENGUPTA, *Stochastic linear programming with chance constraints*, Int. Econ. Rev. 11, 101-116, (1970).

[32] A. SHAPIRO, *Monte Carlo sampling methods*, in Handbook of Stochastic Optimization, A. Ruszczyński and A. Shapiro, eds., Elsevier Science Publishers B.V., Amsterdam, Netherlands, 2003.

[33] ——, *Asymptotics of minimax stochastic programs.* Optimization Online, `http://www.optimization-online.org`, 2006.

[34] A. SHAPIRO AND T. HOMEM-DE MELLO, *A simulation-based approach to two-stage stochastic programming with recourse*, Math. Programming, 81 (1998), pp. 301–325.

[35] G. THOMPSON, W. COOPER, AND A. CHARNES, *Characterizations by chance-constrained programming.*, in Recent Advances math. Program., 113-120, 1963.

[36] B. VERWEIJ, S. AHMED, A. KLEYWEGT, G. NEMHAUSER, AND A. SHAPIRO, *The sample average approximation method applied to stochastic routing problems: A computational study.* Optimization Online, `http://www.optimization-online.org`, 2001.

[37] R.-B. WETS, *Challenges in stochastic programming*, Mathematical Programming, 75 (1996), pp. 115–135.

CURRICULUM VITAE

NAME:	DONGXUE MA
ADDRESS:	Department of Industrial Engineering Speed School of Engineering, University of Louisville Louisville, KY 40292
DOB:	Shenyang Liaoning - January 12, 1986
EDUCATION:	B.S.,Logistics Engineering Beijing University of Posts and Telecommunications 2004 - 2008 M.S., Industrial Engineering University of Louisville 2008 - 2009
AWARDS:	University fellowship University of Louisville August 2008 - present. Scholarship Award Department of Logistic Engineering Beijing University of Posts and Telecommunications , China 2004 - 2008

PROFESSIONAL SOCIETIES:

The Institute For Operations Research and The Management Sciences (INFORMS)
Member, 2008 - present

PRESENTATIONS:

Polynomial Approximation for Two-stage Stochastic Programming
Lijian,Chen and Dongxue, Ma
International Conference on Value Chain Sustainability
October 19 - 21, Louisville,KY (2009).

Polynomial Approximation for Two-stage Stochastic Programming
Lijian,Chen and Dongxue, Ma
INFORMS Annual Meeting
October 11 - 14, San Diego, CA (2009).