A Gradual Non-Convexation Penalty Method for Minimizing VaR

by

Jiong Xi

A thesis
presented to the University of Waterloo
in fulfillment of the
thesis requirement for the degree of
Master of Mathematics
in
Computer Science

Waterloo, Ontario, Canada, 2012

Abstract

This thesis investigates the portfolio optimization problem using Value-at-Risk (VaR) as a risk measure, when m sample scenarios are given. Minimizing VaR of a portfolio is computationally difficult: it is non-convex, non-smooth, and has many local minima. Recently Gaivoronski and Pflug [9] define a quantile-based smoothed VaR function to approximate the original VaR; this smoothed VaR function is then minimized to obtain the minimal VaR portfolio. Unfortunately this method suffers two problems. Firstly, computational cost of minimization is high since each function evaluation requires $O(m^3)$ work, where m is the number of scenarios. Secondly, it is difficult to determine the smooth parameter. We propose a new gradual non-convexation penalty method which can efficiently solve a VaR minimization problem. We first introduce an auxiliary variable and formulate the VaR minimization problem as an optimization problem with a probabilistic constraint, which involves a sum of step functions. A continuously differentiable piecewise quadratic function is used to approximate the step function. An exact penalty method is used to solve the constrained optimization problem. In an attempt to reach the global minimizer, we also use a gradual non-convexation process with the initial problem close to a convex problem. The solution of the kth optimization problem is used as the starting point of the $k+1$th problem. As the indexing parameter increases, the problem becomes more non-convex. Our new method has three advantages. Firstly, our formulation is structurally simpler. Secondly, our method is computationally more efficient since each function evaluation requires $O(m)$ work. Thirdly, we use a gradual non-convexation process in an attempt to track the global minimum; this also avoids the difficulty in choosing the smooth parameter. Both historical and synthetic data are used to test our VaR minimization method. We compare our method with both Uryasev and Rockafellar's CVaR minimization method and Gaivoronski and Pflug's quantile-based smoothed VaR method in terms of VaR, CPU time, and efficient frontiers. We show that our gradual non-convexation penalty method yields better minimal VaR portfolio than the other two methods. In addition, we show that the proposed gradual non-convexation penalty method is computationally much more efficient than Gaivoronski and Pflug's quantile-based smoothed VaR method, especially when the number of scenarios is large.

Acknowledgements

First and foremost, I would like to thank my supervisor, Professor Yuying Li, for her guidance and support throughout the whole process of completing this work. It was Yuying who provided me the opportunity to study in the field of computational finance. I have learned a lot about researching from her during the past two years. This thesis would not have been possible without her guidance, persistent help and encouragement.

I would like to thank my readers, Professor Peter Forsyth and Professor Justin Wan, for taking their time to review my thesis and providing valuable comments and suggestions. I would also like to thank my friends in the Scientific Computation Lab. They make my life in Waterloo such a wonderful experience that I will never forget.

Finally, I would like to thank my family for their endless love, support, and encouragement.

Dedication

This is dedicated to my parents, Jingen Xi and Xuexia Gong, for their love, care and support.

Table of Contents

List of Tables

List of Figures

List of Algorithms

Chapter 1

Introduction

1.1 Motivation

Risk is one of the fundamental concerns in a financial market. It measures the uncertainty of profit or loss in a financial market. Risk management is the activity to identify, assess, control, and allocate risk. The phenomenal growth in trading activities since the late 1960s and the rapid advance in the state of information technology have made great contributions to the fast development of risk management in the financial field [7].

Risk assessment is at the core of modern risk management [6]. In order to assess the risk, we need a mathematical tool to quantify risk. Risk measure is a mapping from the uncertainty of the financial market to a real number. Standard deviation is a classical and widely used risk measure. It measures the deviations from the mean value. It is used by Markowitz in his famous Portfolio Selection Theory [12] to measure the risk of a portfolio. It is also used in the capital asset pricing model (CAPM) [18] and in the option pricing theory [3]. Standard deviation is very important and useful. Standard deviation and mean can fully describe a probability distribution function if it is a normal distribution. However, if the probability distribution function is not a normal distribution, more parameters, such as skewness and kurtosis, are needed to describe the distribution [15].

Value at Risk (VaR) is another very important and widely used risk measure in a financial market. It represents the maximum possible loss with a certain confidence level within a time horizon. Different from standard deviation, VaR can be used for any probability distribution functions and it only concerns the downside risk. The need to aggregate different risks in the financial institution as a whole gave rise to the notion of VaR in the 1970s and 1980s. It is now a standard benchmark for a firm-wide measure of risk [8]. The well known RiskMetrics system developed by JP Morgan uses VaR to measure risk over the whole institution. *Amendment to the Capital Accord to Incorporate Market Risks* specifies that banks using an internal model should compute VaR as a quantitative standard [14]. Down believes that VaR has some significant attractive properties over traditional risk measures: it provides a common consistent measure of risk across different positions and risk factors; it represents risk in a probabilistic way and it is expressed in the simplest and the most easily understood unit of measure [7].

Conditional Value at Risk (CVaR) is a risk measure that is closely related to VaR. It measures the expected value of the loss that exceeds VaR for a specified confidence level. It also measures the downside risk. Instead of using one single loss to represent risk, CVaR takes all the tail distribution into consideration. CVaR is generally regarded as a better risk measure than VaR. CVaR is coherent while VaR is not sub-additive and therefore, not a coherent measure of risk.

Portfolio optimization yields the positions of financial instruments of a portfolio to achieve some objectives such as maximizing mean return and minimizing risk. Markowitz's Portfolio Selection Theory [12] is the pioneer work in this field. He uses standard deviation as a risk measure. More recently, risk measures such as VaR and CVaR are used as a substitution for standard deviation in the portfolio optimization problem. Although CVaR has a better property in terms of coherency than VaR, VaR optimization problem is still very popular both in industry and research for four reasons. Firstly, VaR is widely used and has become an industrial standard. Secondly, VaR is conceptually very simple and easy to understand. Thirdly, no risk measure is dominant over all the other risk measures. Investors who want to control VaR cannot achieve the goal by optimizing CVaR or other risk measures of a portfolio. Fourthly, in the research area, we have already had some accurate and efficient algorithms to find an optimal CVaR portfolio [19]. However, no VaR optimization algorithm that is both accurate and efficient has been found. Moreover, Sarykalin et al. [17] claim that VaR may be better for optimizing portfolios when good models for tails are not available. VaR disregards the hardest to measure events. CVaR may not perform well out of sample when portfolio optimization is run with poorly constructed set of scenarios. Historical data may not give right predictions of future tail events because of mean-reverting characteristics of assets. High returns typically are followed by low returns; hence, CVaR based on history may be quite misleading in risk estimation.

In this thesis, we focus on the portfolio optimization problem using VaR as a risk measure when a finite number of scenarios are given. Minimizing VaR of a portfolio is computationally complex and more difficult than some other portfolio optimization problems such as mean-variance optimization. VaR minimization problem is non-convex, non-smooth, and has many local minima under a finite number of scenarios. A lot of effort has been made to find an efficient minimizing VaR algorithm. Uryasev and Rockafellar [19] propose an efficient algorithm that uses linear programming technique to minimize CVaR of a portfolio and at the same time obtains the corresponding VaR. They claim that this algorithm also yields a low VaR since VaR is always smaller than CVaR. Subsequently, Larsen et al.

[11] propose two heuristic algorithms for VaR optimization that take advantage of efficient CVaR optimization method. The main idea is to start with the minimal CVaR portfolio using an approach proposed by Uryasev and Rockafellar [19] and systematically reduce VaR by solving a series of CVaR optimization problems that are obtained by constraining and discarding scenarios that correspond to large losses. More Recently, Gaivoronski and Pflug [9] propose a quantile-based smoothed VaR method. They reformulate VaR as a quantile of the portfolio loss and use a quantile-based smoothed VaR function (QBSVaR) to replace the original VaR function in the VaR minimization problem. This function filters out the irregularity of the original VaR function. They optimize the quantile-based smoothed VaR function to obtain an optimal VaR portfolio. In addition, Wozabal [20] formulate VaR optimization problem as a difference of convex (D.C.) program and apply the difference of convex algorithm, a generic approximate solution technique for unconstrained D.C. problems, to solve the problem. They also mention that the computational time of the difference of convex algorithm becomes prohibitively long for large data sets. However, among all the proposed methods, we do not see any algorithms that are simple, efficient and accurate. This motivates us to devote our research to the VaR optimization problem.

1.2 Thesis Contributions

In this thesis, we first present a formulation of VaR minimization problem and show the computational difficulty in solving the VaR minimization problem. We also discuss the advantages and disadvantages of two VaR minimization methods, Uryasev and Rockafellar's CVaR minimization method [19] and Gaivoronski and Pflug's quantile-based smoothed VaR (QBSVaR) minimization method [9], respectively. We choose these two methods for comparison because the former is a classical and simple approach for VaR and CVaR minimization problem and the latter uses a smooth technique, which is similar to that is used in our new method.

The CVaR minimization method [19] is an efficient method to solve CVaR minimization problem. However, a minimal CVaR portfolio does not necessarily have a minimal VaR, even though the VaR of a portfolio is always smaller than the CVaR of a portfolio. Gaivoronski and Pflug's quantile-based smoothed VaR (QBSVaR) minimization method [9] uses a smooth approximation function to replace the VaR function in the VaR minimization problem. Their approach is based on the definition that VaR is a quantile of the portfolio losses. This method is complex since it needs to enumerate all possible combinations of choosing k indices from m numbers. A smooth parameter, ϵ, is used to control how accurate the approximation is. However, it is very difficult to choose the smooth parameter and it usually depends on the data. Another drawback of this smooth method is the computational complexity. This method takes $O(m^3)$ time to evaluate the objective function of the optimization problem in addition to evaluating each loss function of the portfolio where m is the number of scenarios.

We propose a new gradual non-convexation penalty (GNCP) method for the VaR minimization problem. We introduce an auxiliary variable α and define VaR_β as the minimum α value which satisfies the constraint that the probability of loss greater than α cannot exceed $1-\beta$. The probabilistic constraint is written as a sum of step functions. We use a piecewise quadratic smooth function to approximate the step function and formulate the VaR minimization problem as an optimization problem with nonlinear constraint which is solved by an exact penalty method. Our new method is conceptually simpler than the quantile-based smoothed VaR method [9]. The computational complexity of our method is also much better than the quantile-based smoothed VaR method [9]. Our method takes $O(m)$ work for function evaluation while the quantile-based smoothed VaR method [9] takes $O(m^3)$ work. Furthermore, our method uses a gradual non-convexation process in an attempt to track the global minimizer. This process is indexed by a parameter ρ, which controls how accurate the approximation is. We gradually increase the parameter and solve the VaR minimization problem by solving a sequence of nonlinearly constrained optimization problems using an exact penalty method. This avoids the difficulty in choosing the smooth parameter.

Computational results are presented to compare the performance between our VaR minimization method and the other two methods in terms of accuracy and time efficiency. Both historical and synthetic data are used in the computational investigations. We show that our new VaR minimization method yields better optimal VaR values for both historical and synthetic data. The efficient frontier of our GNCP method dominates that of the QBSVaR method. In addition, our proposed method is computationally more efficient especially when the number of scenarios, m, is large. The CPU time for our GNCP method is less than linear with respect to m while that for the QBSVaR method is more than quadratic with respect to m.

1.3 Thesis Organization

This thesis is organized as follows: Chapter 2 defines the VaR optimization problem we want to solve in this thesis. We first introduce the classic Mean-Variance portfolio optimization model. Then we provide the formal definition of VaR and CVaR and formulate the VaR optimization model. We also give a simple example to illustrate the computational difficulty in solving the VaR optimization problem.

Chapter 3 reviews Uryasev and Rockafellar's CVaR optimization method [19]. We show how CVaR optimization problem can be formulated as a linear programming problem which can be solved very efficiently. We also discuss the potential problems in using a CVaR optimal portfolio as a VaR optimal portfolio.

Chapter 4 reviews Gaivoronski and Pflug's quantile-based smoothed VaR (QBSVaR) optimization method [9]. We show how the VaR function can be approximated by a smoothed function. Potential problems of the QBSVaR optimization method are also discussed in this chapter.

Chapter 5 proposes our new gradual non-convexation penalty (GNCP) method for the VaR optimization problem. We compare the smooth technique we use with that of quantile-based smoothed VaR (QBSVaR) method [9]. We also analyze the computational cost of our method.

Chapter 6 compares our gradual non-convexation penalty method with the CVaR optimization method [19] and the quantile-based smoothed VaR method [9] in terms of the VaR value achieved and CPU time. We also compare the efficient frontiers obtained from our method with that obtained from the quantile-based smoothed VaR method [9].

Chapter 7 concludes the main contributions we made in this thesis and gives some suggestions for future work.

Chapter 2

Problem Formulation

2.1 Portfolio Optimization

Modern portfolio theory was first introduced by Markowitz in 1952 [12]. The theory is used to solve the asset selection problem. In this formulation, an optimal portfolio is determined to maximize the expected return of the portfolio for a given level of portfolio risk, or equivalently to minimize the portfolio risk for a given level of expected return. Markowitz assumes that asset returns are jointly normal and defines the risk as variance or standard deviation of the return.

Suppose we are given a finite set of assets, $i = 1, 2, \ldots, n$, we want to determine the percentage holding for each asset such that the portfolio meets objectives on return and risk. We can follow Markowitz's theory to form the portfolio, i.e., determine the position of each asset, using the following formulation:

$$\begin{aligned} \min_{\mathbf{x}} \quad & \mathbf{x}^T Q \mathbf{x} \\ \text{s.t.} \quad & \mathrm{E}(\mathbf{x}^T \mathbf{r}) \geq R \\ & \sum_{i=1}^{n} x_i = 1 \\ & x_i \geq 0 \qquad i = 1, 2, \ldots, n \end{aligned} \tag{2.1}$$

where $\mathbf{x} = (x_1, x_2, \ldots, x_n)^T \in \mathbb{R}^n$ is the position of each asset in the portfolio, Q is an $n \times n$ covariance matrix of the asset returns, R is a target level of expected return, $\mathrm{E}(\mathbf{x}^T \mathbf{r}) = \mathbf{x}^T \mathrm{E}(\mathbf{r})$ is the expected return of the portfolio. The constraint $x_i \geq 0$ means asset position can only be non-negative, i.e., no short selling is allowed.

Problem (2.1) is commonly referred to as the Mean-Variance optimization problem. It has become a classical problem in modern portfolio optimization. Many variations have been proposed based on the Mean-Variance optimization model. Equivalently, this problem can be expressed as follows:

$$\begin{aligned} \max_{\mathbf{x}} \quad & E(\mathbf{x}^T \mathbf{r}) \\ \text{s.t.} \quad & \mathbf{x}^T Q \mathbf{x} \leq \bar{R} \\ & \sum_{i=1}^{n} x_i = 1 \\ & x_i \geq 0 \qquad i = 1, 2, \ldots, n \end{aligned} \tag{2.2}$$

where $\bar{R}$ is a target level of risk. Both (2.1) and (2.2) are convex quadratic programming problems. Efficient optimization methods are available to solve these problems.

Conceptually, the Mean-Variance model is very simple. It only needs the expected value and the covariance matrix of the asset returns. However, this model assumes that the return distribution is jointly normal. Usually this assumption is not satisfied in practice.

Even if the underlying return distribution is normal distribution, we can only obtain the sample mean and sample covariance matrix, which are estimations of the true asset mean returns and the true covariance matrix of the returns respectively. The estimation error may give us poor optimal portfolios. Moreover, the return distribution may be skewed and risk cannot be quantified by variance or standard deviation of the return. Considering that VaR and CVaR are risk measures that are suitable for any distribution functions, they can be a good alternative for these situations.

2.2 VaR and CVaR Definitions

2.2.1 VaR Definition

Value at Risk (VaR) is a way to measure the risk of a portfolio. Given a confidence level β, VaR_β of a portfolio is the loss of the portfolio such that with β confidence (e.g. 95%), the portfolio loss will not exceed this value.

Let L be a random loss with probability density function $p(L)$, then the VaR definition can be expressed as follows:

$$\mathrm{VaR}_\beta(L) = \min\{\alpha \in \mathbb{R} : \int_{L\leq\alpha} p(L)\,\mathrm{d}L \geq \beta\}. \tag{2.3}$$

In this thesis, the negative of return, denoted by $f(\mathbf{x},\mathbf{r}) = -\mathbf{x}^T\mathbf{r}$, is referred to as loss, where $\mathbf{x} = (x_1, x_2, \ldots, x_n)^T \in \mathbb{R}^n$ represents the position of each asset in a portfolio and $\mathbf{r} = (r_1, r_2, \ldots, r_n)^T \in \mathbb{R}^n$ is a random vector representing the asset returns.

In the context of portfolio optimization, VaR is a function of asset position $\mathbf{x}$. For a fixed vector $\mathbf{x}$, the portfolio loss $f(\mathbf{x},\mathbf{r})$ is a random variable in $\mathbb{R}$. VaR_β is the value for $f(\mathbf{x},\mathbf{r})$ such that with probability $\beta \in (0,1)$, $f(\mathbf{x},\mathbf{r})$ does not exceed VaR_β. Suppose that the probability density function of $\mathbf{r}$ is $p(\mathbf{r})$. Then the probability of $f(\mathbf{x}, \mathbf{r})$ not exceeding a threshold α can be written as

$$\psi(\mathbf{x}, \alpha) = \int_{f(\mathbf{x},\mathbf{r}) \leq \alpha} p(\mathbf{r}) \, \mathrm{d}\mathbf{r} \tag{2.4}$$

and VaR_β, denoted by $\alpha_\beta(\mathbf{x})$, is given by the following equation:

$$\alpha_\beta(\mathbf{x}) = \min\{\alpha \in \mathbb{R} : \psi(\mathbf{x}, \alpha) \geq \beta\}. \tag{2.5}$$

2.2.2 CVaR Definition

Conditional Value at Risk (CVaR) is a risk measure that is closely related to VaR. For a fixed vector $\mathbf{x}$ representing asset positions, CVaR_β is the conditional expectation of the portfolio loss that exceeds VaR_β. Following the notation in section 2.2.1, CVaR_β, denoted by $\phi_\beta(\mathbf{x})$, can be written as

$$\phi_\beta(\mathbf{x}) = (1 - \beta)^{-1} \int_{f(\mathbf{x},\mathbf{r}) \geq \alpha_\beta(\mathbf{x})} f(\mathbf{x}, \mathbf{r}) p(\mathbf{r}) \, \mathrm{d}\mathbf{r}. \tag{2.6}$$

Figure 2.1 illustrates the definition of VaR and CVaR. The shaded area has $1-\beta$ of the total area under the probability density function. The loss corresponding to the left bound of the shaded area is VaR_β. The expected value of the portfolio loss in the shaded area is CVaR_β. From the definition and the figure, it is easy to see that for any fixed position $\mathbf{x}$ and confidence level β, VaR_β is always no greater than CVaR_β, i.e., $\phi_\beta(\mathbf{x}) \geq \alpha_\beta(\mathbf{x})$.

2.2.3 Coherent Risk Measure

A risk measure can be treated as a mapping from portfolio return, which is a random variable, to a real number. Let $\mathcal{G}$ be the set of all possible portfolio returns. A risk

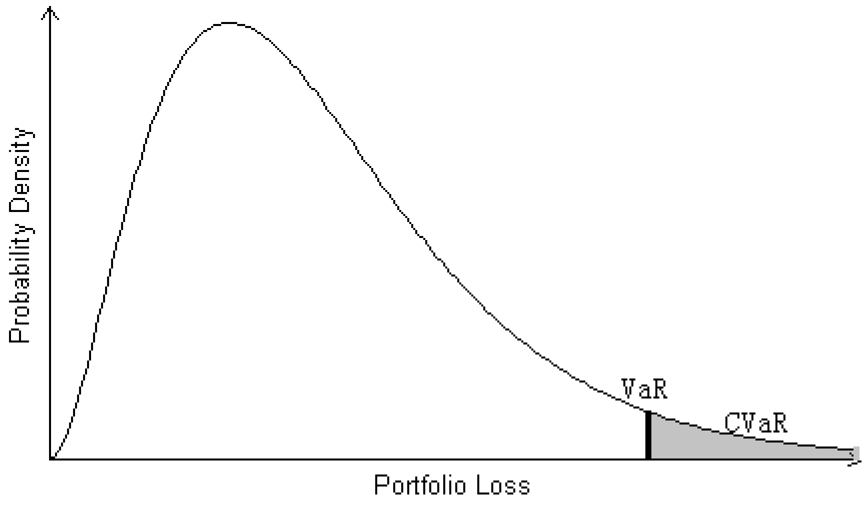

Figure 2.1: Definition of VaR and CVaR

measure $\rho : \mathcal{G} \to \mathbb{R}$ is called a coherent risk measure if it satisfies the following four properties [2]:

1. Translation invariance: for all $X \in \mathcal{G}$ and all $\alpha \in \mathrm{R}^+$, $\rho(X + \alpha) = \rho(X) + \alpha$.

2. Subadditivity: for all X_1 and $X_2 \in \mathcal{G}$, $\rho(X_1 + X_2) \leq \rho(X_1) + \rho(X_2)$.

3. Positivity homogeneity: for all $\lambda \geq 0$ and all $X \in \mathcal{G}$, $\rho(\lambda X) = \lambda\rho(X)$.

4. Monotonicity: for all X and $Y \in \mathcal{G}$ such that $X \leq Y$, $\rho(Y) \leq \rho(X)$.

Pflug [16] shows that CVaR satisfies all the four properties and therefore, it is a coherent risk measure. However, VaR only satisfies properties of translation invariance, positivity homogeneity, and monotonicity. It does not satisfy the property of subadditivity and hence it is not a coherent risk measure.

Although CVaR has better property in terms of coherency than VaR, VaR is still very popular both in industry and research. The problem of the choice between VaR and CVaR, especially in financial risk management, has been quite popular in academic literature. Reasons affecting the choice between VaR and CVaR are based on the differences in mathematical properties, stability of statistical estimation, simplicity of optimization procedures, acceptance by regulators, etc [17]. Sarykalin et al. [17] claim that VaR also has better property than CVaR such as stability of estimation procedures. Because VaR disregards the tail, it is not affected by very high tail losses, which are usually difficult to measure.

2.3 VaR Minimization Problem

In Markowitz's Mean-Variance optimization model, variance is used as a risk measure. Some variations of the model can be made by replacing variance with other risk measures. As mentioned above, VaR is a risk measure. It can be used as a substitution for variance. Replacing variance with VaR in model (2.1) gives us a Mean-VaR optimization model.

Suppose that we are given a finite set of assets, $i = 1, 2, \dots, n$, and each asset i has m sample returns $r_i^{(1)}, r_i^{(2)}, \dots, r_i^{(m)}$. We want to form a portfolio, i.e., determine the position of each asset $\mathbf{x} = (x_1, x_2, \dots, x_n)^T \in \mathbb{R}^n$, such that VaR_β of the portfolio loss is minimized for a target level of expected return R. This problem can be formulated using equation

(2.7) below,

$$\begin{aligned} \min_{\mathbf{x}} \quad & \mathrm{VaR}_\beta(-\mathbf{x}^T\mathbf{r}) \\ \text{s.t.} \quad & \mathrm{E}(\mathbf{x}^T\mathbf{r}) \geq R \\ & \sum_{i=1}^{n} x_i = 1 \\ & x_i \geq 0 \qquad i = 1, 2, \ldots, n. \end{aligned} \tag{2.7}$$

In contrast to the Mean-Variance optimization problem (2.1), Mean-VaR optimization problem (2.7) is very difficult to solve. The objective function is non-smooth, non-convex, and has many local minima. The main contribution of this thesis is that we propose an efficient method to solve Mean-VaR optimization problem. For ease of exposition, we first focus on the difficult part of the problem and we omit, for now, the first constraint on the return in problem (2.7). This gives us the following VaR minimization problem:

$$\begin{aligned} \min_{\mathbf{x}} \quad & \mathrm{VaR}_\beta(-\mathbf{x}^T\mathbf{r}) \\ \text{s.t.} \quad & \sum_{i=1}^{n} x_i = 1 \\ & x_i \geq 0 \qquad i = 1, 2, \ldots, n. \end{aligned} \tag{2.8}$$

We now use a very simple example to show how difficult it is to solve problem (2.8). We use 1,000 daily returns of two stocks, AA and AXP, in Dow Jones stock index from September 4th, 1991 to August 17th, 1995. We want to form a two-asset portfolio that minimizes $\mathrm{VaR}_{0.95}$ of the portfolio. The weight of each asset in the portfolio is written as $\mathbf{x} = (x_1, x_2)^T = (w, 1-w)^T$, $w \in [0, 1]$.

Figure 2.2 shows the $\mathrm{VaR}_{0.95}$ and $\mathrm{CVaR}_{0.95}$ of the portfolio with respect to w. We can see that the VaR function is non-smooth, non-convex, and has many local minima. The VaR optimization problem is generally sensitive to the starting point. Different starting point may yield different local minimizer. However, the CVaR function is convex, smooth

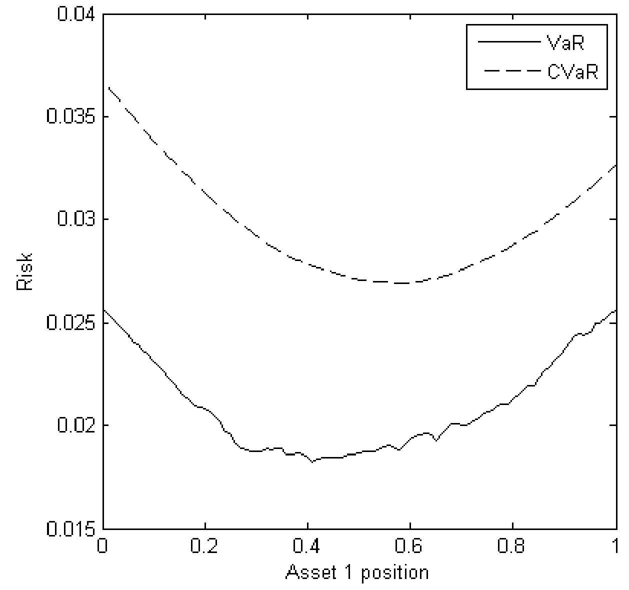

Figure 2.2: $\text{VaR}_{0.95}$ and $\text{CVaR}_{0.95}$ with respect to asset 1 position for a two-asset portfolio

and has only one local minimum, which is also a global minimum. This makes it easy to optimize. Meanwhile, we also see that the VaR function and the CVaR function do not achieve global minimum at the same value of w. This simple example shows that minimizing VaR of a portfolio is computationally challenging. In practice, we can have hundreds and thousands of assets and therefore the problem of minimizing VaR is much more complex than this example.

Chapter 3

CVaR Minimization

Uryasev and Rockafellar [19] propose an efficient CVaR optimization method. Their focus is, however, on minimizing Conditional Value at Risk (CVaR). In addition, they believe that their algorithm also yields a good approximation of a solution for minimizing VaR since they believe that portfolios with low CVaR typically have low VaR as well. In this chapter, we briefly present the CVaR minimization method [19]. We also discuss advantages and disadvantages in using this method to obtain an optimal VaR portfolio. In Chapter 6 we will compare VaR given by the CVaR minimization method with the optimal VaR from our algorithm.

3.1 CVaR Minimization

Similar to a Mean-VaR optimization problem (2.7), a Mean-CVaR optimization problem is also a variation of Mean-Variance optimization problem (2.1), obtained by replacing variance with the CVaR risk measure. It can be formulated as follows:

$$\begin{aligned}
\min_{\mathbf{x}} \quad & \mathrm{CVaR}_\beta(-\mathbf{x}^T\mathbf{r}) \\
\text{s.t.} \quad & \mathrm{E}(\mathbf{x}^T\mathbf{r}) \geq R \\
& \sum_{i=1}^{n} x_i = 1 \\
& x_i \geq 0 \qquad i = 1, 2, \ldots, n.
\end{aligned} \tag{3.1}$$

Since we first want to focus on the VaR minimization problem (2.8), which does not take the constraint on expected portfolio return into consideration, we also omit the first constraint in problem (3.1) for now and consider first the following CVaR minimization problem:

$$\begin{aligned}
\min_{\mathbf{x}} \quad & \mathrm{CVaR}_\beta(-\mathbf{x}^T\mathbf{r}) \\
\text{s.t.} \quad & \sum_{i=1}^{n} x_i = 1 \\
& x_i \geq 0 \qquad i = 1, 2, \ldots, n.
\end{aligned} \tag{3.2}$$

3.2 Solving CVaR Minimization Problems

Uryasev and Rockafellar [19] propose an efficient method to solve CVaR minimization problem (3.2). The key to their method is to define an auxiliary function F_β on $\mathbf{X} \times \mathbb{R}$, where $\mathbf{X} \subseteq \mathbb{R}^n$ is a feasible region of asset positions. The definition of F_β is shown in the equation below.

$$F_\beta(\mathbf{x}, \alpha) = \alpha + (1-\beta)^{-1} \int_{\mathbf{r}\in\mathbb{R}^n} [f(\mathbf{x}, \mathbf{r}) - \alpha]^+ p(\mathbf{r})\, \mathrm{d}\mathbf{r} \tag{3.3}$$

$$[t]^+ = \begin{cases} t & \text{if } t > 0 \\ 0 & \text{if } t \leq 0 \end{cases} \tag{3.4}$$

Uryasev and Rockafellar show that (see Uryasev and Rockafellar [19], Theorem 1 and Theorem 2), under the assumption that the probability density function $p(\mathbf{r})$ does not have jumps, $F_\beta(\mathbf{x}, \alpha)$ is convex and continuously differentiable as a function of α. CVaR_β of the loss associated with any $\mathbf{x} \in \mathbf{X}$ can be determined from the formula

$$\phi_\beta(\mathbf{x}) = \min_{\alpha \in \mathbb{R}} F_\beta(\mathbf{x}, \alpha). \tag{3.5}$$

The set consisting of the values of α for which the minimum is attained, namely

$$A_\beta(\mathbf{x}) = \underset{\alpha \in \mathbb{R}}{\operatorname{argmin}} F_\beta(\mathbf{x}, \alpha), \tag{3.6}$$

is a nonempty, closed, bounded interval (possibly a single point), and the VaR_β of the loss is given by

$$\alpha_\beta(\mathbf{x}) = \text{left endpoint of } A_\beta(\mathbf{x}). \tag{3.7}$$

In particular, one always has

$$\alpha_\beta(\mathbf{x}) \in \underset{\alpha \in \mathbb{R}}{\operatorname{argmin}} F_\beta(\mathbf{x}, \alpha) \text{ and } \phi_\beta(\mathbf{x}) = F_\beta(\mathbf{x}, \alpha_\beta(\mathbf{x})). \tag{3.8}$$

Minimizing the CVaR_β of the loss associated with $\mathbf{x}$ over all $\mathbf{x} \in \mathbf{X}$ is equivalent to minimizing $F_\beta(\mathbf{x}, \alpha)$ over all $(\mathbf{x}, \alpha) \in \mathbf{X} \times \mathbb{R}$, i.e.,

$$\min_{\mathbf{x} \in \mathbf{X}} \phi_\beta(\mathbf{x}) \quad \equiv \min_{(\mathbf{x}, \alpha) \in \mathbf{X} \times \mathbb{R}} F_\beta(\mathbf{x}, \alpha). \tag{3.9}$$

Moreover a pair $(\mathbf{x}^*, \alpha^*)$ achieves the minimum on the right-hand side of (3.9) if and only if $\mathbf{x}^*$ achieves the minimum on the left-hand side and $\alpha^* \in A_\beta(\mathbf{x}^*)$.

Computationally, the integral in definition (3.3) can be approximated by sampling the probability distribution of $\mathbf{r}$ according to its density function $p(\mathbf{r})$. If we have m historical returns of n assets, we can treat them as the sample returns from the return distribution.

Denote the sample returns as $\mathbf{r}^{(1)}, \mathbf{r}^{(2)}, \ldots, \mathbf{r}^{(m)}$, the approximation to $F_\beta(\mathbf{x}, \alpha)$ can be written as

$$\tilde{F}_\beta(\mathbf{x}, \alpha) = \alpha + \frac{1}{m(1-\beta)} \sum_{k=1}^{m} [f(\mathbf{x}, \mathbf{r}^{(k)}) - \alpha]^+. \tag{3.10}$$

Then the CVaR minimization problem (3.2) can be approximated by

$$\begin{aligned} \min_{\mathbf{x},\alpha} \quad & \alpha + \tfrac{1}{m(1-\beta)} \sum_{k=1}^{m} [-\mathbf{x}^T \mathbf{r}^{(k)} - \alpha]^+ \\ \text{s.t.} \quad & \sum_{i=1}^{n} x_i = 1 \\ & x_i \geq 0 \qquad i = 1, 2, \ldots, n. \end{aligned} \tag{3.11}$$

Problem (3.11) can be reduced to a linear programming problem by using auxiliary variables z_k, $k = 1, 2, \ldots, m$:

$$\begin{aligned} \min_{\mathbf{x},\alpha,z} \quad & \alpha + \tfrac{1}{m(1-\beta)} \sum_{k=1}^{m} z_k \\ \text{s.t.} \quad & z_k \geq 0 \qquad k = 1, 2, \ldots, m \\ & \mathbf{x}^T \mathbf{r}^{(k)} + \alpha + z_k \geq 0 \qquad k = 1, 2, \ldots, m \\ & \sum_{i=1}^{n} x_i = 1 \\ & x_i \geq 0 \qquad i = 1, 2, \ldots, n. \end{aligned} \tag{3.12}$$

Many optimization software packages, e.g., MOSEK and CPLEX, can be used to solve problem (3.12) efficiently.

Alexander et al. [1] propose another computationally efficient method for solving problem (3.11). They replace the piecewise linear function $[t]^+$ in equation (3.11) with a continuously differentiable piecewise quadratic approximation function $\tau_\delta(t)$. For a given resolution parameter $\delta > 0$,

$$\tau_\delta(t) = \begin{cases} t & \text{if } t \geq \delta \\ \frac{t^2}{4\delta} + \frac{1}{2}t + \frac{1}{4}\delta & \text{if } -\delta \leq t \leq \delta \\ 0 & \text{if } t \leq -\delta \end{cases} \cdot \tag{3.13}$$

Then the CVaR minimization problem (3.11) can be rewritten as following:

$$\begin{aligned} \min_{\mathbf{x},\alpha} \quad & \alpha + \frac{1}{m(1-\beta)} \sum_{k=1}^{m} \tau_\delta(-\mathbf{x}^T \mathbf{r}^{(k)} - \alpha) \\ \text{s.t.} \quad & \sum_{i=1}^{n} x_i = 1 \\ & x_i \geq 0 \qquad i = 1, 2, \ldots, n. \end{aligned} \tag{3.14}$$

The advantage of this method is that the dimension of problem (3.14) is $O(n)$ instead of $O(m+n)$ as in problem (3.12).

3.3 Problems with Minimization VaR via Minimizing CVaR

The CVaR minimization method [19] is a very accurate and efficient method for the CVaR optimization problem. Considering the fact that VaR_β is always smaller than CVaR_β for any fixed vector $\mathbf{x}$ and β, Uryasev and Rockafellar [19] claim that the optimal CVaR portfolio also has a small VaR value and therefore, the portfolio formed using this method is a good approximation for VaR minimization problem (2.8). Moreover, they show that if the loss associated with each $\mathbf{x}$ is normally distributed, as holds when $\mathbf{r}$ is normally distributed, and $\beta \geq 0.5$, the optimal solution for the CVaR minimization problem (3.1) is also optimal for the VaR minimization problem (2.7) and the Mean-Variance optimization problem (2.1) as well (see Uryasev and Rockafellar [19], Proposition).

However, the problem of using this method to obtain an optimal VaR portfolio is that the probability density function $p(\mathbf{r})$ may not be normal. Moreover, in practice, we may want to solve VaR optimization problems based on sample directly. Even if the probability density function $p(\mathbf{r})$ is normal, the sample returns may differ greatly from the underlying distribution when the sample size, m, is small. In this case, the minimal CVaR$_\beta$ portfolio is not necessarily a minimal VaR$_\beta$ portfolio. A method that minimizes VaR directly may yield better results. In Chapter 6, we will compare VaR provided by the optimal CVaR portfolio with that given by our GNCP VaR minimization method proposed in Chapter 5. We will see that the optimal CVaR portfolios can be significantly suboptimal VaR portfolios.

Chapter 4

A Smoothing Method for Minimizing VaR

More recently, Gaivoronski and Pflug [9] propose a quantile-based smoothed VaR (QB-SVaR) method for VaR optimization problem (2.8). When a finite set of scenarios are given, VaR can be expressed as a k-quantile loss for an integer k. Using this, the VaR function is represented as a weighted combination of the portfolio loss, $\sum_{i=1}^{m} \tilde{c}_i(-\mathbf{x}^T\mathbf{r}^{(i)})$, in which the i-th coefficient, $\tilde{c}_i$, is an indicator of whether the i-th loss scenario, $-\mathbf{x}^T\mathbf{r}^{(i)}$, is the k-quantile loss. Specifically, the i-th indicator coefficient is determined by checking whether there exists a subset of exactly k losses, excluding the i-th loss, such that all these losses do not exceed the i-th loss. This combinatorial search leads to a function whose evaluation has an exponential computational complexity in the number of scenarios, when no smoothing heuristics is applied. They apply a smooth technique on the indicator function and obtain a quantile-based smoothed VaR function, which is an approximation to the original VaR function. Different from the VaR function, the quantile-based smoothed

VaR function is twice continuously differentiable. Therefore it is much easier to solve the quantile-based smoothed VaR minimization problem than to solve the original VaR minimization problem. However, this method is not computationally efficient. Although the smooth technique on the indicator function reduces the time for combinatorial search from exponential time to cubic time, $O(m^3)$, where m is the number of scenarios, the cubic computational complexity for function evaluation is still unsatisfactory.

4.1 Quantile-Based Smoothed VaR Approximation

From the simple example in section 2.3, we can see that although the VaR function with respect to the portfolio positions is non-smooth, non-convex, non-differentiable, and has many local minima, it still has some patterns. Specifically, we can regard the VaR function as a superposition of two components. One is the global component that represents the macrostructure of the VaR function. The other is the local component, introduced by step function in the quantile definition, that is responsible for the small irregularity of the VaR function. The key idea of the quantile-based smoothed VaR function is to extract the global component of the VaR function and filter out the local component.

Gaivoronski and Pflug [9] first express the VaR function as a linear combination of all the sample losses with the coefficients being the sum of some indicator functions. In fact, this expression is exactly the β quantile of the portfolio loss, where β is the confidence level for VaR. Then they rewrite the indicator function as the product of step functions. After that, they use a smooth function to replace the step function in the formulation of the VaR function and get an approximation of the original VaR function. This approximation function is smooth and twice continuously differentiable and we refer it as a quantile-based smoothed VaR (QBSVaR) function.

We now describe their approach in details, following Gaivoronski and Pflug's notation. Let us use the function $M_{[k:m]}(u^1,\ u^2, \dots, u^m)$ to denote the k-th smallest value among $u^1, u^2, \dots, u^m$. Suppose we have a finite number of sample returns $\mathbf{r}^{(1)}, \mathbf{r}^{(2)}, \dots, \mathbf{r}^{(m)}$. For any fixed vector $\mathbf{x}$ that represents the asset positions, the VaR_β of the portfolio can be written as

$$\mathrm{VaR}_\beta = M_{[\lfloor \beta m \rfloor : m]}(-\mathbf{x}^T\mathbf{r}^{(1)}, -\mathbf{x}^T\mathbf{r}^{(2)}, \dots, -\mathbf{x}^T\mathbf{r}^{(m)}). \tag{4.1}$$

The VaR definition they use is slightly different from our definition. According to our VaR definition (2.5), $\lfloor \beta m \rfloor$ should be $\lceil \beta m \rceil$ in equation (4.1). In our computational investigations, we always choose an integer value for βm and therefore, there is no difference between the two definitions.

Let us denote the loss of the portfolio corresponding to the i-th sample return $f_i(\mathbf{x}) = -\mathbf{x}^T\mathbf{r}^{(i)}$. For a given integer k, $0 \le k \le m-1$, and vector $\mathbf{x}$, we define the function

$$G(k, \mathbf{x}) = f_j(\mathbf{x}), \quad j = j(\mathbf{x}) \tag{4.2}$$

such that j satisfies the following two conditions:

1. Inequality $f_j(\mathbf{x}) \le f_i(\mathbf{x})$ is satisfied for at least k functions $f_i(\mathbf{x})$, $i \ne j$.
2. Inequality $f_j(\mathbf{x}) \ge f_i(\mathbf{x})$ is satisfied for at least $m-k-1$ functions $f_i(\mathbf{x})$, $i \ne j$.

The definition of $G(k, \mathbf{x})$ shows that for a finite collection of functions $f_i(\mathbf{x})$, $i = 1, 2, \dots, m$, $G(k, \mathbf{x})$ always equals to the function $f_j(\mathbf{x})$ such that there are at least k functions that are greater than or equal to it and at least $m-k-1$ functions that are smaller than or equal to it, i.e., $G(k, \mathbf{x})$ is the k-th largest loss. Hence, the VaR_β definition in equation (4.1) can be written as

$$\mathrm{VaR}_\beta = M_{[\lfloor \beta m \rfloor : m]}(-\mathbf{x}^T\mathbf{r}^{(1)}, -\mathbf{x}^T\mathbf{r}^{(2)}, \dots, -\mathbf{x}^T\mathbf{r}^{(m)}) = G(m - \lfloor \beta m \rfloor, \mathbf{x}). \tag{4.3}$$

In fact, the function $G(m - \lfloor \beta m \rfloor, \mathbf{x})$ express VaR_β as the β quantile of the portfolio loss. Consequently, the VaR optimization problem (2.8) can be written as

$$\begin{aligned} \min_{\mathbf{x}} \quad & G(m - \lfloor \beta m \rfloor, \mathbf{x}) \\ \text{s.t.} \quad & \sum_{i=1}^{n} x_i = 1 \\ & x_i \geq 0 \qquad i = 1, 2, \ldots, n \end{aligned} \tag{4.4}$$

The next step is to find an approximation function, $G_\epsilon(k, \mathbf{x})$, of the VaR function $G(k, \mathbf{x})$ such that

1. $G_\epsilon(k, \mathbf{x})$ is twice continuously differentiable with respect to $\mathbf{x}$ for all $\epsilon > 0$.

2. $G_\epsilon(k, \mathbf{x}) \to G(k, \mathbf{x})$ as $\epsilon \to 0$.

The function $G_\epsilon(k, \mathbf{x})$ is a smooth approximation of the original quantile-based VaR function $G(k, \mathbf{x})$, and therefore we refer it as the quantile-based smoothed VaR (QBSVaR) function of the portfolio.

Gaivoronski and Pflug [9] first use a linear combination of the loss functions $f_i(\mathbf{x})$ to represent $G(k, \mathbf{x})$ with all the coefficients being a function of $\mathbf{x}$. And then they use some smooth functions to approximate the coefficients and obtain the QBSVaR function $G_\epsilon(k, \mathbf{x})$. Before we give the expressions for the coefficients and the QBSVaR function, we introduce some notations that are used in [9]:

- M^i: the set of all integers from 1 to m except i, i.e., $M^i = \{1, 2, \ldots, m\} \backslash \{i\}$

- Λ_k^i: an arbitrary subset of M^i which contains exactly k elements

- $X(\Lambda_k^i)$: a subset of $\mathbb{R}^n$ associated with the set Λ_k^i such that

$$X(\Lambda_k^i) = \{\mathbf{x} : f_i(\mathbf{x}) \leq f_j(\mathbf{x}) \text{ for } j \in \Lambda_k^i, f_i(\mathbf{x}) \geq f_j(\mathbf{x}) \text{ for } j \in M^i \backslash \Lambda_k^i\}$$

- Θ_i^k: the family of all different sets Λ_k^i

Gaivoronski and Pflug [9] show that (see Gaivoronski and Pflug [9], Lemma 1) the VaR function $G(k, \mathbf{x})$ can be expressed using the following form:

$$G(k, \mathbf{x}) = \frac{1}{C(\mathbf{x})} \sum_{i=1}^{m} c_i(\mathbf{x}) f_i(\mathbf{x}) \tag{4.5}$$

where

$$C(\mathbf{x}) = \sum_{i=1}^{m} c_i(\mathbf{x}) \tag{4.6}$$

$$c_i(\mathbf{x}) = \sum_{\Lambda_k^i \in \Theta_k^i} \mathbb{I}_{X(\Lambda_k^i)}(\mathbf{x}) \tag{4.7}$$

$$\mathbb{I}_{X(\Lambda_k^i)}(\mathbf{x}) = \prod_{j \in \Lambda_k^i} \mathbb{I}_-\big(f_i(\mathbf{x}) - f_j(\mathbf{x})\big) \prod_{j \in M^i \setminus \Lambda_k^i} \mathbb{I}_-\big(f_j(\mathbf{x}) - f_i(\mathbf{x})\big) \tag{4.8}$$

$$\mathbb{I}_-(z) = \begin{cases} 1 & \text{if } z \le 0 \\ 0 & \text{if } z > 0 \end{cases}. \tag{4.9}$$

The indicator function $\mathbb{I}_{X(\Lambda_k^i)}(\mathbf{x})$ in equation (4.8) equals to 1 only if the index set Λ_k^i is chosen such that all the losses corresponding to the indices in the set are no less than $f_i(\mathbf{x})$ and all the losses corresponding to the indices not in the set are no greater than $f_i(\mathbf{x})$. This means $f_i(\mathbf{x})$ is the k-th largest loss among all the losses. Hence $c_i(\mathbf{x}) > 0$ only when the i-th loss $f_i(\mathbf{x})$ is VaR_β of the portfolio.

Although the loss function $f_i(\mathbf{x}) = -\mathbf{x}^T \mathbf{r}^{(i)}$ is a linear function (thus twice continuously differentiable), the indicator function $\mathbb{I}_{X(\Lambda_k^i)}(\mathbf{x})$ in equation (4.8) is not continuous and this makes the VaR function $G(k, \mathbf{x})$ not continuously differentiable. Gaivoronski and Pflug [9] use the following statement to get the QBSVaR function $G_\epsilon(k, \mathbf{x})$ (see Gaivoronski and Pflug [9], Theorem 2). Let $\varphi_\epsilon(z)$ be a function defined for $z \in \mathbb{R}$ and $\epsilon \ge 0$ such that

1. $\varphi_\epsilon(z)$ is twice continuously differentiable for $\epsilon > 0$
2. $\varphi_\epsilon(z) \to 1$ as $\epsilon \to 0$ for any fixed $z \leq 0$
3. $\varphi_\epsilon(z) \to 0$ as $\epsilon \to 0$ for any fixed $z > 0$
4. $\varphi_\epsilon(z) \geq 0$ for all $\epsilon \geq 0, z$ and $\varphi_\epsilon(z) \geq \chi_0$ for some $\chi_0 > 0$ and all $\epsilon \geq 0, z \leq 0$.

Then the function $G_\epsilon(k, \mathbf{x})$ defined below,

$$G_\epsilon(k, \mathbf{x}) = \frac{1}{C_\epsilon(\mathbf{x})} \sum_{i=1}^{m} c_i^\epsilon(\mathbf{x}) f_i(\mathbf{x}), \tag{4.10}$$

where

$$C_\epsilon(\mathbf{x}) = \sum_{i=1}^{m} c_i^\epsilon(\mathbf{x}), \tag{4.11}$$

$$c_i^\epsilon(\mathbf{x}) = \sum_{\Lambda_k^i \in \Theta_k^i} \prod_{j \in \Lambda_k^i} \varphi_\epsilon\big(f_i(\mathbf{x}) - f_j(\mathbf{x})\big) \prod_{j \in M^i \setminus \Lambda_k^i} \varphi_\epsilon\big(f_j(\mathbf{x}) - f_i(\mathbf{x})\big), \tag{4.12}$$

is twice continuously differentiable for all $\epsilon > 0$, assuming $\{f_i(\mathbf{x})\}$ are twice continuously differentiable. In addition, $G_\epsilon(k, \mathbf{x}) \to G(k, \mathbf{x})$ as $\epsilon \to 0$ for any fixed $\mathbf{x}$.

Gaivoronski and Pflug [9] choose the following cubic spline function for $\varphi_\epsilon(z)$:

$$\varphi_\epsilon(z) = \begin{cases} 1 & \text{if } z \leq 0 \\ 1 - \frac{16}{3\epsilon^3} z^3 & \text{if } 0 \leq z \leq \frac{\epsilon}{4} \\ \frac{5}{6} + \frac{2}{\epsilon} z - \frac{8}{\epsilon^2} z^2 + \frac{16}{3\epsilon^3} z^3 & \text{if } \frac{\epsilon}{4} \leq z \leq \frac{3\epsilon}{4} \\ \frac{16}{3} - \frac{16}{\epsilon} z + \frac{16}{\epsilon^2} z^2 - \frac{16}{3\epsilon^3} z^3 & \text{if } \frac{3\epsilon}{4} \leq z \leq \epsilon \\ 0 & \text{if } z \geq \epsilon \end{cases}. \tag{4.13}$$

As $\epsilon \to 0$, $\varphi_\epsilon(z)$ will approach to the step indicator function $\mathbb{I}_-(z)$ in equation (4.9).

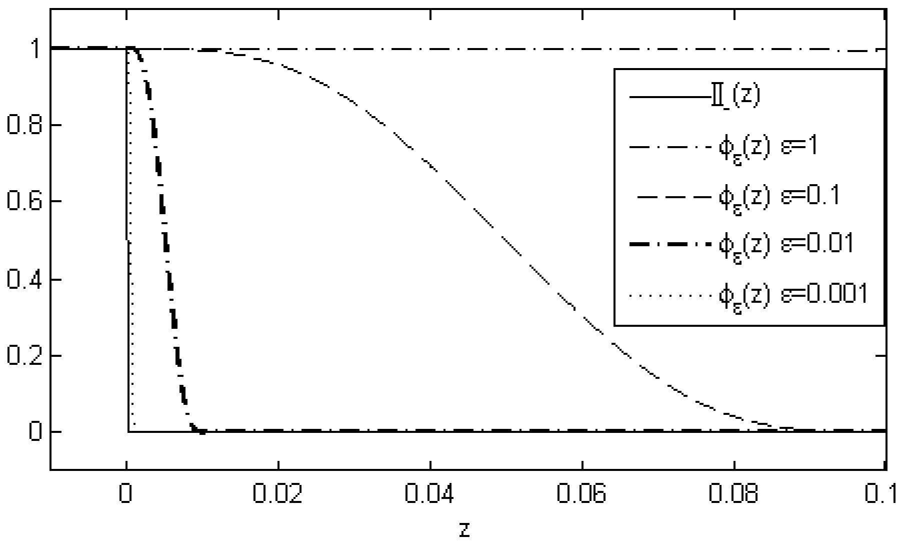

Figure 4.1: Plots of $\mathbb{I}_{-}(z)$ and $\varphi_\epsilon(z)$, $\epsilon = 1, 0.1, 0.01, 0.001$

.

Figure 4.1 shows plots of $\mathbb{I}_{-}(z)$ and $\varphi_\epsilon(z)$ when $\epsilon = 1, 0.1, 0.01, 0.001$. The solid line is the plot of $\mathbb{I}_{-}(z)$. The thin dash-dot line at the top is $\varphi_\epsilon(z)$ when $\epsilon = 1$. The dashed line below it is $\varphi_\epsilon(z)$ when $\epsilon = 0.1$. The thick dash-dot line is $\varphi_\epsilon(z)$ when $\epsilon = 0.01$. The dotted line is $\varphi_\epsilon(z)$ when $\epsilon = 0.001$. The smooth function $\varphi_\epsilon(z)$ only differs from the step indicator function $\mathbb{I}_{-}(z)$ in the interval $[0, \epsilon]$. We see that as ϵ becomes smaller, this gap will be smaller.

The key idea of the QBSVaR function $G_\epsilon(k, \mathbf{x})$ is to rewrite the indicator function $\mathbb{I}_{X(\Lambda_k^i)}$ as a product of some step indicator functions $\mathbb{I}_{-}(z)$, and then use a smooth function $\varphi_\epsilon(z)$ to approximate the step indicator function $\mathbb{I}_{-}(z)$. Equation (4.12) shows that in order to compute $c_i^\epsilon(\mathbf{x})$, we need to enumerate all the possible combinations of choosing k indices from m numbers. Thus computational cost grows exponentially. Gaivoronski and Pflug [9] show that the computational cost can be reduced if we choose the smooth function $\varphi_\epsilon(z)$

according to equation (4.13). We will present this analysis in section 4.2.

The parameter ϵ in the QBSVaR function plays an important role. The QBSVaR function $G_\epsilon(k, \mathbf{x})$ is defined such that $G_\epsilon(k, \mathbf{x}) \to G(k, \mathbf{x})$ as $\epsilon \to 0$. Meanwhile it is easy to verify that $G_\epsilon(k, \mathbf{x}) \to \frac{1}{m}\sum_{i=1}^{m} f_i(\mathbf{x})$ as $\epsilon \to \infty$ which is the average loss of the portfolio and is twice continuously differentiable. Hence the choice of the parameter ϵ determines how accurate and how smooth the approximation is. If the value of ϵ is large, the corresponding QBSVaR function is smoother and easy to optimize. Unfortunately, the approximation loses a lot of information of the original VaR function. On the other hand, if ϵ is small, the approximation is very accurate while the VaR minimization problem is difficult to solve with many local minimizers. The choice of ϵ is a tradeoff between difficulty and accuracy.

4.2 A Quantile-Based Smoothed VaR Minimization Method

Replacing VaR_β with the quantile-based smoothed VaR function $G_\epsilon(m - \lfloor \beta m \rfloor, \mathbf{x})$ in e-quation (2.8), we have the following quantile-based smoothed VaR minimization problem:

$$\begin{aligned} \min_{\mathbf{x}} \quad & G_\epsilon(m - \lfloor \beta m \rfloor, \mathbf{x}) \\ \text{s.t.} \quad & \sum_{i=1}^{n} x_i = 1 \\ & x_i \geq 0 \qquad i = 1, 2, \ldots, n \end{aligned} \tag{4.14}$$

Computationally, the QBSVaR minimization method [9] consists of the following three steps:

Step 1 Construct a QBSVaR function $G_\epsilon(m - \lfloor\beta m\rfloor, \mathbf{x})$ that extracts the global component of the VaR function. Use the QBSVaR function to formulate the VaR optimization problem as equation (4.14).

Step 2 Solve the smooth approximation problem (4.14) by some standard software for nonlinear programming problems.

Step 3 Postprocess the computed solution obtained in Step 2.

One of the key parts of the QBSVaR minimization method [9] is Step 1. Gaivoronski and Pflug [9] mention that general approximation techniques, such as spline approximation can be used for the QBSVaR function. However, the problem is that those approximations may not be efficient in computational time, since it grows fast with n, the dimension of the portfolio. Considering that we need to evaluate the QBSVaR function many times in the procedure of optimization, these methods are impractical. As mentioned in section 4.1, the computational cost is still large to compute the coefficients $c_i^\epsilon(\mathbf{x})$ as we need to enumerate all the possible combinations of choosing k indices from m numbers. Gaivoronski and Pflug [9] show that (see Gaivoronski and Pflug [9], Lemma 3 and Theorem 5) if the function $\varphi_\epsilon(z)$ is chosen according to equation (4.13), the approximation function $G_\epsilon(k, \mathbf{x})$ can be equivalently expressed as follows:

$$G_\epsilon(k, \mathbf{x}) = \frac{1}{C_\epsilon(\mathbf{x})} \sum_{\{i : |G(k,\mathbf{x}) - f_i(\mathbf{x})| \leq \epsilon\}} c_i^\epsilon(\mathbf{x}) f_i(\mathbf{x}). \tag{4.15}$$

This means only a few coefficients $c_i^\epsilon(\mathbf{x})$ need to be computed for each fixed vector $\mathbf{x}$. This reduces the time it needs to compute the approximation function $G_\epsilon(k, \mathbf{x})$. Furthermore, they provide an algorithm for which the number of additional arithmetic operations, N_G, necessary for computation of $G_\epsilon(k, \mathbf{x})$ for any fixed k, ϵ and $\mathbf{x}$ can be estimated as follows:

$$N_G \leq (1 + s_m)m^3 \tag{4.16}$$

where m is the number of functions $f_i(\mathbf{x})$, $s_m \to 0$ as $m \to \infty$, $s_m \leq 15$ and s_m does not depend on the dimension of vector $\mathbf{x}$. Although it is a great progress compared to exponential time, it is still cubic in m for just one function evaluation.

For Step 2, Gaivoronski and Pflug [9] use **fmincon** subroutine from MATLAB Optimization Toolbox to solve the nonlinear quantile-based smoothed VaR optimization problem (4.14). This subroutine uses an iterative method to solve a constrained nonlinear optimization problem. For each iteration, the computational cost consists of three parts. The first part is function evaluation, including evaluating the objective function and the constraints. For problem (4.14) the constraint is linear and the time for this part mainly depends on the time for objective function evaluation, which is $O(m^3)$ in addition to evaluating each loss function. The second part is gradient and hessian evaluation. Since the objective function is complex, it is not easy to give an analytic form for the gradient and the hessian of the objective function $G_\epsilon(k, \mathbf{x})$. The **fmincon** subroutine uses finite difference to approximate the gradient and calculates the hessian by a dense quasi-Newton approximation. The time complexity is $O(nm^3)$. The third part is to solve an n-by-n linear system of equation, which takes $O(n^3)$ time. In practice, the number of samples m is usually greater than the number of assets n. Therefore, the computational time for each iteration is $O(nm^3)$.

The solution we obtain from Step 2 is a local minimizer of the QBSVaR function. Step 3 in the QBSVaR minimization method [9] tries to make further improvement on the solution obtained from Step 2 as follows. Let us denote $\mathbf{x}_{\text{SVaR}}$ the optimal solution obtained from Step 2. Gaivoronski and Pflug [9] propose to solve the following optimization problem for

Step 3:

$$\begin{aligned}
\min_{\mathbf{x}} \quad & f_j(\mathbf{x}) \\
\text{s.t.} \quad & f_i(\mathbf{x}) - f_j(\mathbf{x}) \leq 0, i \in \Lambda \\
& f_i(\mathbf{x}) - f_j(\mathbf{x}) \geq 0, i \notin \Lambda \\
& \sum_{i=1}^{n} x_i = 1 \\
& x_i \geq 0 \qquad i = 1, 2, \ldots, n
\end{aligned} \tag{4.17}$$

where j is the index such that $\text{VaR}_\beta = f_j(\mathbf{x}_{\text{SVaR}})$, Λ is an arbitrary set of $\lfloor \beta m \rfloor$ indices i that satisfy $f_i(\mathbf{x}_{\text{SVaR}}) \leq f_j(\mathbf{x}_{\text{SVaR}})$. This linear programming program fixes the index that achieves VaR_β and the indices below and over the VaR_β value and optimizes the quantile function. Gaivoronski and Pflug [9] believe that the improvement of VaR is relatively small using this method and therefore the postprocessing procedure is optional.

4.3 Problems with the Quantile-Based Smothed VaR Minimization Method

The quantile-based smoothed VaR minimization method [9] is a reasonable approximation method for the VaR minimization problem (2.8). However, this method has three problems.

The first problem of the QBSVaR minimization method [9] is the computational time complexity. Gaivoronski and Pflug [9] show that the computational time complexity to evaluate the approximation function $G_\epsilon(k, \mathbf{x})$ is $O(m^3)$ where m is the number of return samples, plus the time to evaluate the loss functions $f_i(\mathbf{x})$, $i = 1, 2, \ldots, m$. As we mentioned in section 4.2, we need to evaluate $G_\epsilon(k, \mathbf{x})$ many times for the optimization procedure. This is costly even if the the time for each function evaluation is $O(m^3)$. Especially when the number of scenarios, m, is large, the computational time complexity is unsatisfying.

Our new VaR minimization method, which will be proposed in Chapter 5, has much better performance in terms of efficiency than the QBSVaR minimization method [9]. In Chapter 6, we will also present the results that compare the CPU time of the QBSVaR minimization method and our new VaR minimization method on both historical and synthetic data sets.

The second problem of the QBSVaR minimization method [9] is that it is difficult to compute the first and the second order derivatives of the objective function. When solving an optimization problem, the gradient and the hessian of the objective function are often needed. Optimization solvers usually compute the gradient and the hessian by using some numeric methods such as finite difference. If the analytic forms of the gradient and the hessian of the QBSVaR function are provided, the optimization problem can be solved more efficiently and accurately. Gaivoronski and Pflug [9] do not take advantage of the analytic forms of the gradient and the hessian of the QBSVaR function in their QBSVaR minimization method. In fact, although the QBSVaR function $G_\epsilon(k, \mathbf{x})$ is twice continuously differentiable, the analytic form of the gradient and the hessian are very complicated since the coefficient $c_i^\epsilon(\mathbf{x})$ is the sum of a few products of $m-1$ functions $\varphi_\epsilon(z)$. Therefore, the **fmincon** subroutine needs to numerically approximate the gradient and the hessian of the objective function. The complexity of function evaluation makes gradient and hessian evaluation difficult and potentially unnecessarily expensive.

The third problem of the QBSVaR minimization method is the difficulty in choosing the parameter ϵ. As we mentioned previously, the choice of the parameter ϵ determines how accurate the approximation is and how difficult it is to solve the optimization problem. It is a tradeoff between accuracy and difficulty. Gaivoronski and Pflug [9] do not give any suggestion on how to choose the parameter ϵ. In practice, the choice of parameter depends on the data set that we use and it is usually hard to determine whether the value of the parameter is a good choice for the data set. As the size of the problem increases, the choice for the parameter becomes even more difficult since the VaR function is higher dimensional and has more local minima. On one hand, the parameter, ϵ, should be large enough to make the optimization problem easy to solve. However, on the other hand, large ϵ means losing much information of the VaR function. And also the effect of reducing the time to evaluate the QBSVaR function $G_\epsilon(k, \mathbf{x})$ induced by equation (4.15) is less prominent. The conflict becomes intensive for large scale problems.

Chapter 5

A Gradual Non-Convexation Penalty Method for Minimizing VaR

In this chapter, we propose a new method for the VaR minimization problem (2.8). Similar to Gaivoronski and Pflug [9], we also use a smooth function to approximate the step function in our formulation. However, the difference is that we introduce an auxiliary variable α and express the VaR_β of a portfolio as the minimum loss value α such that the probability of a loss exceeding α is no greater than $(1-\beta)$. Hence, the VaR definition becomes a probabilistic constraint of an optimization problem. This avoids combinatorial complexity in using quantile as a definition of VaR. A gradual non-convexation optimization process is introduced to facilitate tracking the global minimizer. The computational complexity of function and constraint evaluation for this method is $O(m)$, where m is the number of scenarios. This technique is similar to Coleman et al. [5] for the index tracking problem of finding a portfolio that minimizes a chosen measure of the index tracking error. We now apply this technique to the VaR minimization problem (2.8).

5.1 An Equivalent VaR Optimization Formulation

Let $\mathbb{I}_+(z)$ denote a step indicator function defined by the following equation:

$$\mathbb{I}_+(z) = \begin{cases} 1 & \text{if } z > 0 \\ 0 & \text{otherwise} \end{cases}. \tag{5.1}$$

Suppose we are given n assets, each of which has m sample returns. Let $\mathbf{r}^{(i)} \in \mathbb{R}^n$ denote the i-th sample return of assets in the portfolio, $1 \leq i \leq m$. Let $\mathbf{x} = (x_1, x_2, \ldots, x_n)^T \in \mathbb{R}^n$ denote the percentage positions of the assets in a portfolio. Then the VaR optimization problem (2.8) can be formulated as follows:

$$\begin{aligned} \min_{\mathbf{x},\alpha} \quad & \alpha \\ \text{s.t.} \quad & \sum_{i=1}^{m} \mathbb{I}_+(-\mathbf{x}^T\mathbf{r}^{(i)} - \alpha) \leq (1-\beta)m \\ & \sum_{i=1}^{n} x_i = 1 \\ & x_i \geq 0 \qquad i = 1, 2, \ldots, n. \end{aligned} \tag{5.2}$$

In the optimization problem (5.2), the objective function $\alpha \in \mathbb{R}$ is an auxiliary variable. If the i-th loss, $-\mathbf{x}^T\mathbf{r}^{(i)}$, is greater than α, the step function $\mathbb{I}_+(-\mathbf{x}^T\mathbf{r}^{(i)} - \alpha)$ equals 1, otherwise, it equals 0. Therefore the left-hand side of the first constraint in problem (5.2), the probabilistic constraint, is the number of sample losses that are greater than α. This constraint means the number of sample losses that are greater than α cannot exceed $(1-\beta)m$. Hence, the minimum α value satisfying the probability constraint yields VaR_β of a portfolio.

Theorem 1. *Under the assumption that βm is an integer, optimization problem* (4.4) *and optimization problem* (5.2) *are equivalent in the sense that they give the same minimum* VaR_β *of the portfolio.*

Proof. First, as we mentioned before, the VaR definition of Gaivoronski and Pflug [9] is slightly different from our VaR definition. The two definitions are equivalent when $\lfloor \beta m \rfloor = \lceil \beta m \rceil$, i.e., βm is an integer. In our proof, we assume that βm is an integer.

Let $f_i(\mathbf{x}) = -\mathbf{x}^T \mathbf{r}^{(i)}, i = 1, 2, \ldots, m$ be m losses of the portfolio in increasing order, $f_i(\mathbf{x}) \le f_{i+1}(\mathbf{x})$. We need to prove that for any fixed vector $\mathbf{x} \in \mathbb{R}^n$ such that $\sum_{i=1}^n x_i = 1$ and $x_i \ge 0, i = 1, 2, \ldots, n$, the optimal solution α^* for problem (5.2) is equal to the β quantile of the portfolio, i.e., $\alpha^* = f_{\beta m}(\mathbf{x})$.

Let $\bar{\alpha} = f_{\beta m}(\mathbf{x})$. We first show that $(\bar{\alpha}, \mathbf{x})$ is a feasible solution for problem (5.2). Let j be the smallest index such that $f_j(\mathbf{x}) > \bar{\alpha}$. Since $f_i(\mathbf{x})$ is in increasing order, $j \ge \beta m + 1$. Then $m - (j-1) \le m - \beta m$. There are $m - (j-1)$ losses that are greater than $\bar{\alpha}$. Hence, $\sum_{i=1}^m \Lambda_+(-\mathbf{x}^T \mathbf{r}^{(i)} - \bar{\alpha}) = m - (j-1) \le m - \beta m = (1-\beta)m$, i.e., the probabilistic constraint in equation (5.2) is satisfied and $(\bar{\alpha}, \mathbf{x})$ is a feasible solution for problem (5.2).

Secondly, for any $\alpha < \bar{\alpha}$, we need to show that $(\alpha, \mathbf{x})$ is not feasible. Let l be the largest index such that $f_l(\mathbf{x}) \le \alpha$. Then $l \le \beta m - 1$. There are $m - l$ losses that are greater than α. Hence $\sum_{i=1}^m \Lambda_+(-\mathbf{x}^T \mathbf{r}^{(i)} - \alpha) = m - l \ge m + 1 - \beta m > (1-\beta)m$. The probabilistic constraint in equation (5.2) does not hold and $(\alpha, \mathbf{x})$ is not feasible.

Therefore, for any fixed feasible vector $\mathbf{x}$, $(\alpha^*, \mathbf{x})$ is an optimal solution for problem (5.2) where $\alpha^* = f_{\beta m}(\mathbf{x})$, i.e. problem (4.4) and problem (5.2) are equivalent. $\square$

Compared to the quantile-based VaR optimization problem (4.4), our VaR optimization formulation (5.2) is much simpler. Both the objective function and the constraint can be evaluated in linear time rather than cubic time. However, the first constraint is a summation of m step functions, which are non-smooth and non-convex. Hence the optimization problem (5.2) is still difficult to solve. Similar to the QBSVaR method [9], we want to use

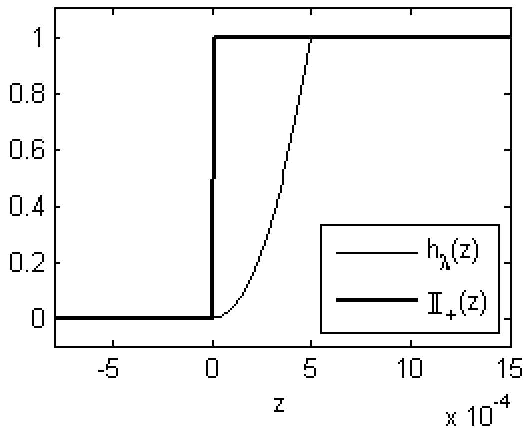

Figure 5.1: Plots of $h_\lambda(z)$ and $\mathbb{I}_+(z)$, $\lambda = 4 \times 10^6$

.

a smooth function to approximate the step function and make it easier to solve problem (5.2). We will explain our method in detail in section 5.2.

5.2 A Smoothed Approximation Function

First, we know that the step function $\mathbb{I}_+(z)$ is not continuous at $z = 0$. We use the following continuous function $h_\lambda(z)$ to approximate the step function $\mathbb{I}_+(z)$:

$$h_\lambda(z) = \begin{cases} 1 & \text{if } z > \frac{1}{\sqrt{\lambda}} \\ \lambda z^2 & \text{if } 0 < z \leq \frac{1}{\sqrt{\lambda}} \\ 0 & \text{if } z \leq 0 \end{cases} . \tag{5.3}$$

Figure 5.1 shows plots of $\mathbb{I}_+(z)$ and $h_\lambda(z)$ respectively. The thick line is the plot of the step function $\mathbb{I}_+(z)$. The thin line is plot of the approximation function $h_\lambda(z)$ when

$\lambda = 4\times 10^6$. We can see that $h_\lambda(z)$ is equal to $\mathbb{I}_+(z)$ outside the interval $[0, \frac{1}{\sqrt{\lambda}}]$. While inside the interval, we use a quadratic function which is equal to 0 at $z = 0$ and is equal to 1 at $z = \frac{1}{\sqrt{\lambda}}$ to approximate $\mathbb{I}_+(z)$. Hence for any fixed parameter $\lambda \in \mathbb{R}^+$, the function $h_\lambda(z)$ is continuous for any $z \in \mathbb{R}$. The functions $h_\lambda(z)$ and $\mathbb{I}_+(z)$ only differ in the interval $[0, \frac{1}{\sqrt{\lambda}}]$. In order to be a good approximation for $\mathbb{I}_+(z)$, the parameter, λ, should be sufficiently large.

The function $h_\lambda(z)$ is continuous but not differentiable at $z = \frac{1}{\sqrt{\lambda}}$. Hence it is still not suitable for typical optimization software. We now define a new function $g_\lambda(z;\rho)$, indexed by a parameter $\rho > 0$, to approximate $h_\lambda(z)$ as follows:

$$g_\lambda(z;\rho) = \begin{cases} 1 & \text{if } z \geq \gamma \\ 1 - \frac{\rho}{2}(z-\gamma)^2 & \text{if } \kappa \leq z \leq \gamma \\ \lambda z^2 & \text{if } 0 \leq z \leq \kappa \\ 0 & \text{if } z \leq 0 \end{cases} \tag{5.4}$$

where

$$\gamma = \sqrt{\frac{2}{\rho} + \frac{1}{\lambda}}, \kappa = \frac{1}{\lambda\gamma}, \tag{5.5}$$

$\rho > 0$ is a parameter of the approximation. The function $g_\lambda(z;\rho)$ is a piecewise quadratic function. Inside each interval, $(-\infty, 0)$, $(0, \kappa)$, (κ, γ), and (γ, ∞), $g_\lambda(z;\rho)$ is differentiable. Let $g'_+(z;\rho)$ and $g'_-(z;\rho)$ denote the right and the left limit of the function $g(z;\rho)$ respectively. When $z = 0$, $g'_+(0;\rho) = 2\lambda z|_{z=0} = 0 = g'_-(0;\rho)$. When $z = \gamma$, $g'_-(\gamma;\rho) = -\rho(z-\gamma)|_{z=\gamma} = 0 = g'_+(\gamma;\rho)$. When $z = \kappa$, $g'_-(\kappa;\rho) = 2\lambda z|_{z=\kappa} = 2\lambda\kappa = \frac{2}{\gamma}$, $g'_+(\kappa;\rho) = -\rho(z-\gamma)|_{z=\kappa} = -\rho(\kappa-\gamma) = -\frac{2}{\gamma^2 - \frac{1}{\lambda}}(\frac{1}{\lambda\gamma} - \gamma) = \frac{2}{\gamma}$, hence $g'_-(\kappa;\rho) = g'_+(\kappa;\rho)$. We have shown that at the endpoints of the interval, $z = 0$, $z = \kappa$, and $z = \gamma$, $g_\lambda(z;\rho)$ is also differentiable and the derivative is continuous. Therefore, for any fixed $\rho \in \mathbb{R}^+$, the function $g_\lambda(z;\rho)$ is continuously differentiable at any point $z \in \mathbb{R}$.

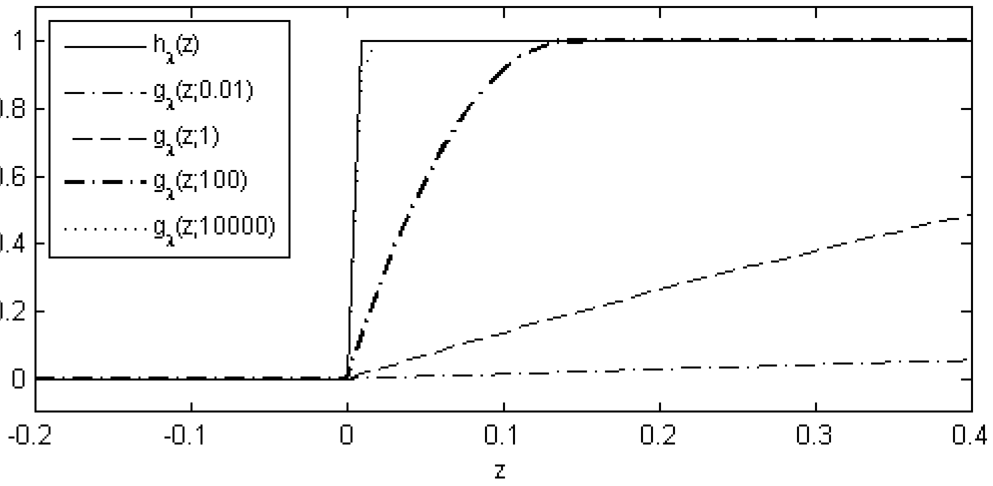

Figure 5.2: Plots of $h_\lambda(z)$ and $g_\lambda(z;\rho)$, $\lambda = 4 \times 10^6$, $\rho = 0.01, 1, 100, 10000$

.

A function $f : \mathbb{R}^n \to \mathbb{R}$ is called quasiconvex (or unimodal) if its domain and all its sublevel sets

$$S_\zeta = \{z \in \mathbf{dom}\ f | f(z) \leq \zeta\},$$

for $\zeta \in \mathbb{R}$, are convex (Boyd and Vandenberghe [4]). For a fixed $\rho \in \mathbb{R}^+$, the function $g_\lambda(z;\rho)$ is a real value function on $\mathbb{R}$. Since $g_\lambda(z;\rho)$ is non-decreasing, each sublevel sets of it is an interval, i.e., a convex set. Therefore, $g_\lambda(z;\rho)$ is a quasiconvex function for any fixed $\rho \in \mathbb{R}^+$.

Figure 5.2 shows plots of $h_\lambda(z)$ and $g_\lambda(z;\rho)$ with $\lambda = 4 \times 10^6$ and $\rho = 0.01, 1, 100, 10000$ respectively. The thin dash-dot line at the bottom is $g_\lambda(z;0.01)$. The dashed line above it is $g_\lambda(z,1)$. The thick dash-dot line in the middle is $g_\lambda(z;100)$. The dotted line is $g_\lambda(z;10000)$. The solid line at the top is $h_\lambda(z)$. We can see that as $\rho \to \infty$, $g_\lambda(z;\rho)$ approaches to $h_\lambda(z)$. In Figure 5.2, $g_\lambda(z;10000)$ is very close to $h_\lambda(z)$ and they only differ in a very small interval

$[\kappa, \gamma]$ to the right of $z = 0$. Actually, it is easy to verify that $\kappa \to 0$ and $\gamma \to +\infty$ as $\rho \to 0$, $\kappa \to \frac{1}{\sqrt{\lambda}}$ and $\gamma \to \frac{1}{\sqrt{\lambda}}$ as $\rho \to +\infty$. This means that when ρ is very small, the interval $[\kappa, \gamma]$ occupies almost the whole range of $\mathbb{R}^+$ and $g_\lambda(z;\rho)$ is close to a linear function in this interval. When the value of ρ increases, the length of the interval $[\kappa, \gamma]$ will shrink to zero and $g_\lambda(z;\rho)$ will approach to $h_\lambda(z)$.

Using $g_\lambda(z;\rho)$ to replace $\mathbb{I}_+(z)$ in problem (5.2), we get the following optimization problem which is an approximation of the VaR minimization problem (2.8).

$$\begin{aligned} \min_{\mathbf{x},\alpha} \quad & \alpha \\ \text{s.t.} \quad & H_\lambda(\mathbf{x};\rho) \le (1-\beta) \\ & \sum_{i=1}^{n} x_i = 1 \\ & x_i \ge 0 \qquad i = 1, 2, \ldots, n, \end{aligned} \tag{5.6}$$

where

$$H_\lambda(\mathbf{x};\rho) = \frac{1}{m} \sum_{i=1}^{m} g_\lambda(-\mathbf{x}^T \mathbf{r}^{(i)} - \alpha; \rho). \tag{5.7}$$

When the parameter ρ is small, problem (5.6) is similar to a linear programming problem and thus close to a convex optimization problem. As we gradually increase ρ, problem (5.6) gradually becomes non-convex. The optimal α when ρ is sufficiently large gives optimal VaR_β of a portfolio. In the next section, we will show how to solve this problem.

5.3 Solving Optimization Problems

5.3.1 Exact Penalty Methods

An exact penalty method can be used to solve a constrained optimization problem. The key idea of a penalty function method is to eliminate a constraint by penalizing constraint violation in the objective function appropriately. For example, the nonlinearly constrained problem (5.6) can be solved using the following exact penalty method:

$$\min_{\mathbf{x}\in\Omega,\alpha} \quad \alpha + \mu \cdot \max\left(H_\lambda\left(\mathbf{x};\rho\right) - (1-\beta), 0\right) \tag{5.8}$$

where $\Omega = \{\mathbf{x} \in \mathbb{R}^n | \sum_{i=1}^{n} x_i = 1, x_i \geq 0 \text{ for } i = 1, 2, \ldots, n\}$.

The parameter, μ, in problem (5.8) is an exact penalty parameter corresponding to the probabilistic constraint in problem (5.6). It should be sufficiently large. When the probabilistic constraint in problem (5.6) is satisfied, the penalty term does not have any impact on the objective function in problem (5.8). On the other hand, if the constraint is not satisfied, the penalty term $\mu \cdot \max\left(H_\lambda\left(\mathbf{x};\rho\right) - \left(1-\beta\right), 0\right)$ is large and this makes the objective function of problem (5.8) large. A suitable choice of the penalty parameter is problem dependent in general. It can be adaptively determined as follows. We first pick a reasonable large value of the parameter μ and obtain the corresponding optimal solution $\mathbf{x}^*$. Then we check whether the nonlinear constraint $H_\lambda(\mathbf{x}^*;\rho) \leq (1-\beta)$ holds. If it does, the optimal solution $\mathbf{x}^*$ of problem (5.8) is also an optimal solution of problem (5.6). If the constraint does not hold, we need to increase the value of μ and solve problem (5.8) again. We repeat the above procedure until we find an optimal solution of problem (5.8) such that the nonlinear constraint holds.

5.3.2 A Gradual Non-Convexation Penalty Method

Section 5.3.1 describes a way to solve the optimization problem (5.6) for a fixed value of ρ. The parameter ρ in the definition of $g_\lambda(z;\rho)$ has similar effect as the parameter ϵ in the QBSVaR method [9]. It controls how smooth and how accurate the approximation functions is. If ρ is small, the approximation function $g_\lambda(z;\rho)$ is very smooth and the optimization problem (5.6) is easy to solve while the approximation is not accurate. If ρ is large, the approximation function is more accurate while the corresponding optimization problem (5.6) is difficult to solve. From the analysis in section 4.3, we see that it is hard to pick a suitable value for the smoothing parameter ϵ in the QBSVaR method [9]. Our smoothing function has the same problem to choose the the value of the parameter ρ. In order to avoid this problem, we gradually increase the value of ρ. Initially, ρ is very small; we get an optimal solution for problem (5.6) using the exact penalty method. Then we increase the value of ρ and use the optimal solution from the previous step as a starting point to solve the problem (5.6) again. We repeat this process until the interval $[\kappa, \gamma]$ is small enough and there is no loss $-\mathbf{x}^T\mathbf{r}^{(i)}$ such that $z = -\mathbf{x}^T\mathbf{r}^{(i)} - \alpha$ falls in this interval. For a sufficiently small $\rho > 0$, the optimization problem (5.6) is close to a convex programming problem. As we gradually increase ρ, the optimization problem (5.6) gradually becomes non-convex. We call this a gradual non-convexation penalty (GNCP) method for minimizing VaR. Algorithm 5.1 describes this computational process in details.

In our subsequent computations, we choose $\lambda = 4\times10^6$, $\alpha_0 = 0$ and $\mathbf{x}_0 = \frac{1}{n}(1, 1, \ldots, 1)^T$. According to the condition, the length of the interval $[\kappa, \gamma]$ is small enough and there is no $z = -\mathbf{x}^T\mathbf{r}^{(i)} - \alpha$ in this interval when the algorithm terminates. This means the current optimal solution is also optimal for the optimization problem that uses $h_\lambda(z)$ instead of $g_\lambda(z; \rho)$ as a substitution of $\Lambda_+(z)$ in equation (5.2) since $g_\lambda(z; \rho)$ and $h_\lambda(z)$ only differ in the interval $[\kappa, \gamma]$. Therefore this algorithm gives a good approximation solution for the original VaR minimization problem (2.8). In our subsequent computations, ρ is usually around $10^8 \sim 10^9$ when the algorithm terminates.

On the surface, it seems that our gradual non-convexation method is computationally complex since it needs to solve a sequence of optimization problems. Actually, this is not the case. Initially, the optimization problem (5.6) is close to a convex programming

Algorithm 5.1: Gradual Non-Convexation Penalty Method for Minimizing VaR

```
Input: $m$ sample returns for $n$ assets, $\mathbf{r}^{(i)} \in \mathbb{R}^n, i = 1, 2, \ldots, m$; starting point
        $(\alpha_0, \mathbf{x}_0)$
Output: A computed solution $\mathbf{x}$ for problem (2.8)
begin
    Choose a large value for $\lambda$ ;
    $k \longleftarrow 0$ ;
    $\rho_k \longleftarrow 1 \times 10^{-5}$ ;
    $\chi \longleftarrow \{i | -\mathbf{x}_k^T \mathbf{r}^{(i)} - \alpha_k \in [\kappa, \gamma], i = 1, 2, \ldots, m\}$ ;
    while $\chi$ is not empty or $|\gamma - \kappa| > 1 \times 10^{-4}$ do
        $k \longleftarrow k + 1$ ;
        $\rho_k \longleftarrow 10\rho_{k-1}$ ;
        solve problem (5.6) using the exact penalty method (5.8) with $\rho = \rho_k$ and
        starting point $(\alpha_{k-1}, \mathbf{x}_{k-1})$ to obtain the computed solution $(\alpha_k, \mathbf{x}_k)$ ;
    end
    return $\mathbf{x}_k$ ;
end
```

problem and can be solved efficiently. A global optimal solution is likely computed for this problem. As the parameter ρ increases, the optimization problem (5.6) becomes more non-convex and our exact penalty method may only find a local minimum. However, since the consecutive problems are similar due to the fact that ρ is updated gradually, it is likely that our starting point is close to a local minimum and only a few iterations are needed for the optimization procedure to get a local minimum if we use the optimal solution for the previous iteration as a starting point. Moreover, this local minimum is likely close to a global minimum. Therefore, this algorithm does not take excessive number of optimization iterations even though it needs to solve a sequence of optimization problems.

We now use a simple example to illustrate our gradual non-convexation penalty (GNCP) method for minimizing VaR. We use the same 1000 historical samples of the two stock allocation example in section 2.3. Let us consider the function $H_\lambda(\mathbf{x};\rho) = \frac{1}{m}\sum_{i=1}^{m} g_\lambda(-\mathbf{x}^T\mathbf{r}^{(i)} - \alpha;\rho)$ on the left-hand side of the probabilistic constraint in problem (5.6). Note that in the penalty problem (5.8), the penalty term $\mu \cdot \max\left(H_\lambda\left(\mathbf{x};\rho\right) - (1-\beta), 0\right)$ is the main non-convex challenging component in the objective function. We want to show how the function $H_\lambda(\mathbf{x};\rho)$ changes as the parameter ρ approaches to infinity.

First, we fix $\alpha = 0.02$ and draw plots of $H_\lambda(\mathbf{x};\rho)$ with respect to x_1, the first weight of the asset positions $\mathbf{x} = [x_1, 1-x_1]^T$. Figure 5.3 shows plots for different values of ρ. The thick dashed line at the bottom is $H_\lambda(\mathbf{x};10^1)$. It is almost a straight line. The thin dashed line above it is $H_\lambda(\mathbf{x};10^3)$. It is smooth and appears to be convex. The thick dash-dot line in the middle is $H_\lambda(\mathbf{x};10^4)$. It becomes more non-convex and has some local minima. The thin dash-dot line is $H_\lambda(\mathbf{x};10^5)$. The dotted line is $H_\lambda(\mathbf{x};10^7)$. These two functions have more local minima. We also plot the original function $H_o(\mathbf{x}) = \frac{1}{m}\sum_{i=1}^{m}\mathbb{I}_+(-\mathbf{x}^T\mathbf{r}^{(i)} - \alpha)$. It is the solid line at the top in Figure 5.3. We can see that $H_\lambda(\mathbf{x};10^7)$ is very close to the original function $H_o(\mathbf{x})$. As Figure 5.3 illustrates, $H_o(\mathbf{x})$ function is a global convex curve

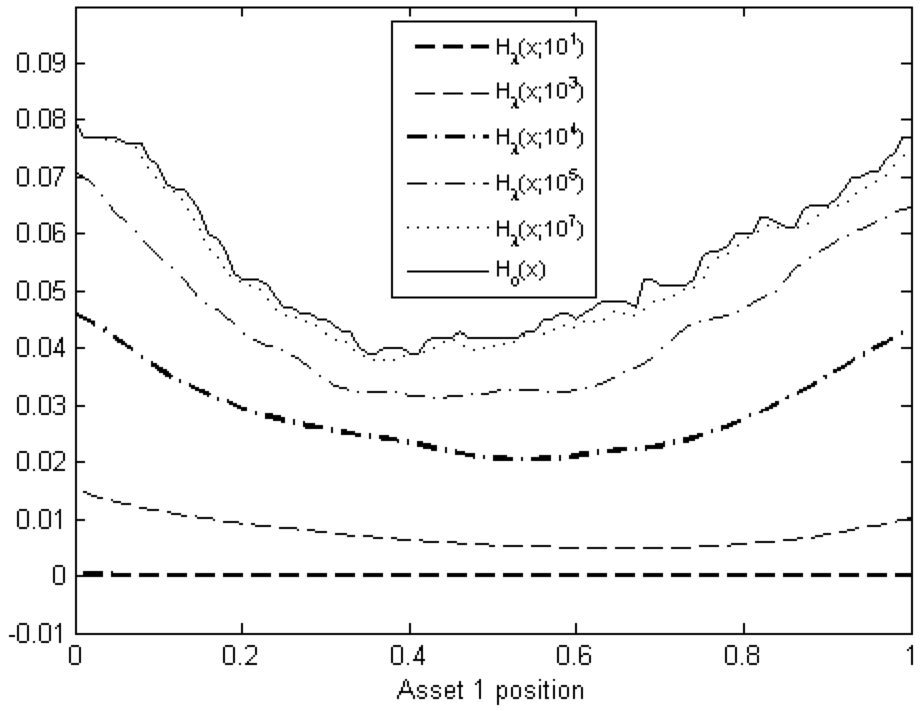

Figure 5.3: Plots of $H_o(\mathbf{x})$ and its approximations $H_\lambda(\mathbf{x};\rho)$ with respect to asset1 position of a two asset portfolio, $\lambda = 4 \times 10^6$, $\alpha = 0.02$, $\rho = 10^1, 10^3, 10^4, 10^5, 10^7$

.

superimposed with many small variations which create multiple local minimizers. The global minimizer is in a neighborhood of the minimizer of this global convex curve. When ρ is small, the function $H_\lambda(\mathbf{x})$ approximates $H_o(\mathbf{x})$ in a global perspective. Meanwhile, the optimization problem is easy to solve and we are likely to get a global minimum. As ρ increases, the function becomes non-convex and is hard to solve. While the computed solution from the previous iteration is a good starting point to obtain a local minimum, which is likely a global minimum for the next optimization problem with ρ_{k+1}.

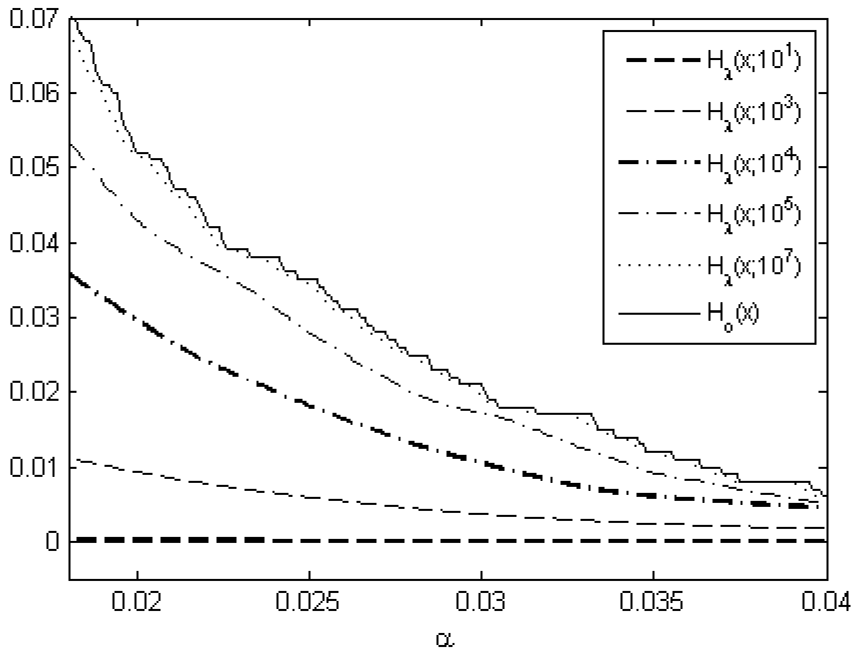

Figure 5.4: Plots of $H_o(\mathbf{x})$ and its approximations $H_\lambda(\mathbf{x};\rho)$ with respect to α of a two asset portfolio, $\lambda = 4 \times 10^6$, $\mathbf{x} = [0.2, 0.8]^T$, $\rho = 10^1, 10^3, 10^4, 10^5, 10^7$

.

We also illustrate dependence relationship of the function $H_\lambda(\mathbf{x};\rho)$ with respective to α. We fix the asset position at $\mathbf{x} = [0.2, 0.8]^T$ and draw plots of $H_\lambda(\mathbf{x};\rho)$ with respect to α for different values of ρ. Figure 5.4 shows plots for $\rho = 10^1$, 10^3, 10^4, 10^5, and 10^7 and the one for the original function $H_o(\mathbf{x})$ respectively from bottom to top. We have similar observations as Figure 5.3. As ρ increases, $H_\lambda(\mathbf{x};\rho)$ as a function of α will approach to a non-smooth function.

We note that the function $H_\lambda(\mathbf{x};\rho)$ changes with respect to both asset positions $\mathbf{x}$ and α simultaneously. However, these two figures give us an intuitive idea how our proposed algorithm works and why it yields a good approximation for minimizing VaR.

Now we come back to Mean-VaR optimization problem (2.7). This optimization problem has one more linear constraint compared to optimization problem (2.8). The linear constraint can be easily integrated into the optimization problem (2.7) by an optimization solver with little increase in computational complexity. Hence, our gradual non-convexation penalty method can also be applied to problem (2.7).

5.4 Comparisons with the QBSVaR Method [9]

Our GNCP method has three advantages over the QBSVaR method [9]. Firstly, our method is computationally more efficient. As mentioned in section 4.3, the QBSVaR method [9] takes $O(m^3)$ extra time to evaluate the approximation function $G_\epsilon(k, \mathbf{x})$. However, for our method, we only need to evaluate the function, $g_\lambda(z; \rho)$, m times and compute summation after we evaluate each loss function $f_i(\mathbf{x}) = -\mathbf{x}^T \mathbf{r}^{(i)}$. It takes only $O(m)$ work to evaluate the function $H_\lambda(\mathbf{x}; \rho) = \frac{1}{m} \sum_{i=1}^{m} g_\lambda(-\mathbf{x}^T \mathbf{r}^{(i)} - \alpha; \rho)$. This is a significant advantage over the QBSVaR method [9].

Secondly, our method attempts to track the global minimizer via a gradual non-convex process. In the QBSVaR method [9], the parameter ϵ is quite difficult to choose and the value of ϵ determines how good the approximation is. In our method, however, the smoothing parameter ρ is gradually increased until the approximation function is sufficiently accurate. We do not need to choose a specific value for ρ. Moreover, the time needed to solve a sequence of optimization problems using our exact penalty method does not increase excessively since we always have a good starting point for each optimization problem.

Last but not least, our method is not sensitive to the starting point for optimization. For optimization problems that have many local minima, the starting point can have a major effect on the result of the optimization. Different starting points may yield different local minima. It is not easy to choose a good starting point for optimization problems in the QBSVaR method [9]. However, our method solves a nearly convex programming problem at the beginning, which has only global minimum and is not sensitive to the starting point for optimization. Even though the following optimization problems are non-convex and may be sensitive to their starting points, our method is robust and, as a whole, is not sensitive to the starting point.

Chapter 6

Computational Comparisons

In this chapter, we present computational results to compare our gradual non-convexation penalty (GNCP) method of minimizing VaR with the CVaR minimization method [19] and the quantile-based smoothed VaR (QBSVaR) method [9]. We compare the results in terms of the computed VaR of the optimal portfolio, the CPU time, and the efficient frontiers. Both historical and synthetic data are used for the computational comparisons. All the tests are done on a 2.91GHz PC that has an AMD Athlon 64 X2 Dual CPU and 3.25 GB RAM.

6.1 Comparisons on Historical Data

In this section, we present the computational comparisons on the historical data. We compare the computed VaR of our GNCP method with that of the CVaR minimization method [19] and the QBSVaR method [9] respectively. We also test the effect of the starting point of our GNCP method. The historical data set, denoted as DS1, contains 10 year historical daily returns (2513 smaples) of 30 stocks in Dow Jones stock index from September 4th, 1991 to September 4th 2001. The names of 30 stocks are listed in Table A.1 in Appendix A.

6.1.1 Suboptimality of VaR from CVaR Minimization

In this section, we compare the VaR from computed portfolio using our GNCP method with that of using CVaR minimization method [19] introduced in Chapter 3. We use the historical data set DS1 for this investigation. For each method, we first compute the optimal asset positions $\mathbf{x}^*$ using the optimization method. Then we compute the corresponding portfolio loss $-\mathbf{x}^{*T}\mathbf{r}^{(i)}$, $i = 1, 2, \ldots, m$, and the VaR_β and CVaR_β of the portfolio based on the loss samples. We use the VaR computed by GNCP method as the benchmark and compare relative difference of VaR computed by CVaR minimization method [19] against it. Specifically we report

$$RD_{VaR}^{C} = \frac{\text{VaR}_{\text{minCVaR}} - \text{VaR}_{\text{GNCP}}}{|\text{VaR}_{\text{GNCP}}|} 100\%. \tag{6.1}$$

Similarly, we report the relative difference of CVaR computed by CVaR minimization method [19] against that computed by our GNCP method.

$$RD_{CVaR} = \frac{\text{CVaR}_{\text{minCVaR}} - \text{CVaR}_{\text{GNCP}}}{|\text{CVaR}_{\text{GNCP}}|} 100\% \tag{6.2}$$

Table 6.1 shows the VaR of the optimal portfolio computed by GNCP method and CVaR minimization method [19] respectively, and the relative difference for different number of assets, n, and different number of sample returns, m, with $\beta = 95\%$. A positive number indicates degradation of minimizing CVaR method over GNCP method. Table 6.2 shows the CVaR of the optimal portfolio computed by each method and the relative difference for

m	n	VaR		RD_{VaR}^{C}
		GNCP	min CVaR	
300	10	0.01061	0.011869	11.87%
300	20	0.0087491	0.011251	28.60%
300	30	0.0074684	0.0089423	19.75%
500	10	0.010772	0.010822	4.64%
500	20	0.0087856	0.010155	15.59%
500	30	0.0075459	0.0088468	17.24%
1000	10	0.010478	0.011192	6.81%
1000	20	0.0089188	0.0097212	9.00%
1000	30	0.0075692	0.0085919	13.51%
2000	10	0.013762	0.014524	5.54%
2000	20	0.012347	0.012703	2.88%
2000	30	0.01039	0.011651	12.14%

Table 6.1: VaR and relative difference of minimizing VaR and minimizing CVaR portfolio, $\beta = 95\%$

different number of assets, n, and different number of sample returns, m, with $\beta = 95\%$. We also report the results for $\beta = 90\%$ in Appendix B. We can see that our VaR minimization method always yields smaller VaR of the portfolio. The VaR obtained using our GNCP method is on average more than 10% better than that obtained using CVaR minimization method [19]. The relative difference ranges from 2.88% to 28.60%. On the other hand, the negative relative difference RD_{CVaR} in Table 6.2 shows that the CVaR of the CVaR minimization method [19] is smaller than that of the VaR minimization method. We can

m	n	CVaR		RD_{CVaR}
		GNCP	min CVaR	
300	10	0.019276	0.017353	-1.44%
300	20	0.017606	0.015874	-9.84%
300	30	0.01436	0.012492	-13.01%
500	10	0.0172	0.016303	-5.22%
500	20	0.015132	0.01451	-4.11%
500	30	0.013938	0.011812	-15.25%
1000	10	0.016429	0.016049	-2.31%
1000	20	0.014818	0.014445	-2.52%
1000	30	0.01325	0.01219	-8.00%
2000	10	0.021632	0.021229	-1.86%
2000	20	0.01991	0.019211	-3.51%
2000	30	0.018082	0.017156	-5.12%

Table 6.2: CVaR and relative difference of minimizing VaR and minimizing CVaR portfolio, $\beta = 95\%$

also see that the CVaR obtained using our GNCP method is less than 5% worse than that obtained using the CVaR minimization method in half of the cases. This may be due to the fact that CVaR function is flatter than the VaR function.

Figure 6.1 shows the $\text{VaR}_{0.95}$ using a rolling window for both VaR minimization method and CVaR minimization method. We use $n = 30$ assets in data set DS1 and set the window size $m = 300$. We first run the optimization algorithm on the first m returns of the assets. Then we move the test window one step forward and run the optimization algorithm on

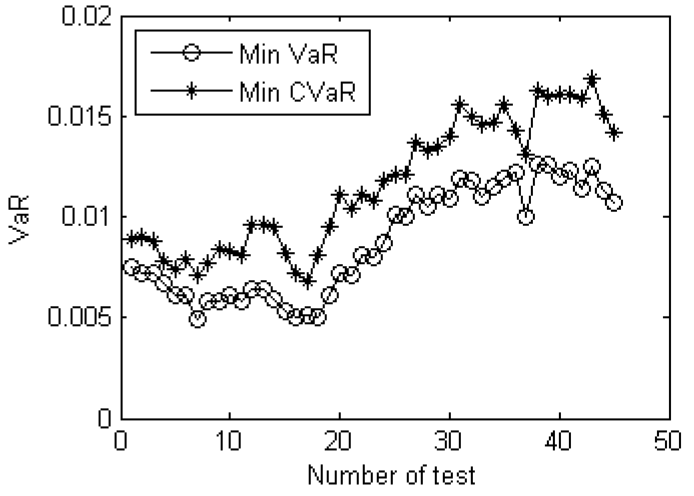

Figure 6.1: Rolling window test for VaR of minimizing VaR and minimizing CVaR, $\beta = 95\%$, $m = 300$,$n = 30$,$sp = 50$

.

the sample returns in the window. The step size sp is set to be 50. We continue moving the window until it reaches the end of the returns in the data set. The star in Figure 6.1 represents the VaR of the optimal CVaR portfolio. The circle represents the VaR of the optimal VaR portfolio. The figure shows that for each test, the VaR minimization method yields smaller VaR than the CVaR minimization portfolio. We also test on returns with different number of assets, n, different window size, m, and different confidence level, β. The results are all similar.

The above test results verify that CVaR optimal portfolio does not necessarily gives us minimal VaR portfolio. These are sound evidence for the need to develop a VaR minimization method that minimizes VaR directly rather than minimizing CVaR.

6.1.2 Effect of Initial Points for GNCP

In section 5.4, we mention that our gradual non-convexation penalty method is not sensitive to the initial point of the optimization algorithm. In this section, we want to test the impact of the initial point on our VaR minimization method. Different from the QBSVaR method [9], the initial point of our GNCP method has one more variable, α, in addition to the the variables for the asset positions, $\mathbf{x}$. We fix $\alpha = 0$ and try two different initial points on the asset position $\mathbf{x}_0$ and compare the VaR we obtain from our GNCP method using the historical data set DS1. The first initial point $(\frac{1}{n})$ is an equal weight portfolio $\mathbf{x}_0 = \frac{1}{n} \times \text{ones}(n, 1)$. The second initial point $(\mathbf{x}^*_{\text{CVaR}})$ is the optimal CVaR portfolio we obtain from the CVaR minimization method [19].

Table 6.3 shows $\text{VaR}_{0.95}$ and $\text{VaR}_{0.9}$ for different number of assets, n, and different number of returns, m. We can see that the VaR of the portfolio corresponding to two initial points have no significant difference. The results show that our GNCP method is not sensitive to the initial point. Starting our algorithm from an optimal CVaR portfolio does not give us portfolio with better optimal VaR. For all the remaining computational investigations, we will use the equal weight initial point for our VaR minimization method.

6.1.3 Comparison Between GNCP and QBSVaR

In this section, we compare our gradual non-convexation penalty (GNCP) method with the quantile-based smoothed VaR (QBSVaR) method [9] in terms of computed VaR on the historical data set DS1. We compute the VaR of the minimal VaR portfolio for different number of assets, n, and different number of returns, m. In order to access numerically the improvement obtained using our GNCP method, we report the relative difference defined

m	n	$\mathrm{VaR}_{0.95}$		$\mathrm{VaR}_{0.9}$	
		$\frac{1}{n}$	$\mathbf{x}^*_{\mathrm{CVaR}}$	$\frac{1}{n}$	$\mathbf{x}^*_{\mathrm{CVaR}}$
300	10	0.01061	0.010958	0.0079865	0.0082694
300	20	0.0087491	0.0087733	0.0072237	0.0073217
300	30	0.0074684	0.0074303	0.0051924	0.0053269
500	10	0.010772	0.010086	0.0078666	0.0078336
500	20	0.0087856	0.0090869	0.0072434	0.0080066
500	30	0.0075459	0.0080684	0.0056612	0.0058597
1000	10	0.010478	0.010247	0.0076282	0.0075285
1000	20	0.0089188	0.0087112	0.0071986	0.0068093
1000	30	0.0075692	0.0076763	0.0056484	0.0056494
2000	10	0.013762	0.013913	0.010026	0.0099563
2000	20	0.012347	0.011696	0.0086397	0.0086409
2000	30	0.01039	0.01044	0.0075813	0.0076355

Table 6.3: Effect of starting points for GNCP method

as follows:

$$RD^{Q}_{VaR} = \frac{\mathrm{VaR}_{\mathrm{QBSVaR}} - \mathrm{VaR}_{\mathrm{GNCP}}}{|\mathrm{VaR}_{\mathrm{GNCP}}|}100\%. \qquad (6.3)$$

Positive relative difference indicates degradation of QBSVaR method over GNCP method.

Table 6.4 shows $\mathrm{VaR}_{0.95}$ and the relative difference for different number of assets, n, and different number of sample returns, m. For QBSVaR method [9], we report results for four different values, 0.01, 0.001, 0.0001, and 0.00001, for the smooth parameter ϵ. We also report $\mathrm{VaR}_{0.90}$ and the relative difference on data set DS1 in Table C.1 in Appendix C. We have the following observations from the test results.

m	n	VaR_{GNCP}	RD_{VaR}^{Q}			
			$\epsilon = 0.01$	$\epsilon = 0.001$	$\epsilon = 0.0001$	$\epsilon = 0.00001$
300	10	0.01061	24.27%	6.66%	8.03%	7.82%
300	20	0.0087491	24.45%	6.17%	4.17%	8.43%
300	30	0.0074684	35.96%	15.11%	12.83%	17.78%
500	10	0.010772	4.91%	0.05%	-5.70%	1.58%
500	20	0.0087856	13.06%	6.27%	2.77%	2.63%
500	30	0.0075459	20.71%	13.47%	10.24%	12.28%
1000	10	0.010478	9.76%	13.56%	3.22%	3.99%
1000	20	0.0089188	9.24%	14.65%	4.61%	7.45%
1000	30	0.0075692	18.93%	21.03%	14.17%	19.47%
2000	10	0.013762	7.06%	5.27%	4.56%	4.40%
2000	20	0.012347	4.73%	4.41%	10.26%	8.16%
2000	30	0.01039	10.31%	11.31%	12.27%	9.96%

Table 6.4: VaR of minimizing VaR portfolio using GNCP method and the relative difference compared to the QBSVaR method on historical data, $\beta = 95\%$

1. The smooth parameter, ϵ, of the QBSVaR method [9] is very difficult to choose. Typically, a smaller ϵ tends to yield better VaR of the portfolio. However, when ϵ is very small, e.g., $\epsilon = 0.0001$ for data set DS1, decreasing ϵ does not necessarily produce better VaR of the VaR optimal portfolio.

2. Compared with the QBSVaR method [9], our GNCP method yield smaller VaR of the VaR optimal portfolio with the exception of one entry $m = 500$, $n = 10$, $\epsilon = 0.0001$. On average, the VaR obtained using our GNCP method is 5$\sim$10% better than that

obtained using QBSVaR with the maximum improvement 35.96%. We also note that when the number of assets, n, is large ($n = 30$), the VaR obtained using our GNCP method is significantly better than that obtained using the QBSVaR method [9].

In conclusion, our GNCP method usually yields optimal VaR portfolio with smaller VaR compared to the QBSVaR method [9]. The test results also show the difficulty in choosing the smooth parameter, ϵ, in the QBSVaR method [9]. Our GNCP method avoids the procedure to choose smooth parameter. This is also a great advantage of our method.

6.2 Comparisons on Synthetic Data

For this section, we present the computational comparisons on a synthetic data set. This synthetic data set has more sample returns and more assets. We compare the computed VaR of our GNCP method with that of QBSVaR method [9]. We also compare the CPU time of the two methods.

6.2.1 Synthetic Data Set

The data set we use in this section, denoted as DS2, is a synthetic data set, generated from multivariate jump models with random model parameters. This data set contains $M = 100000$ returns of $N = 1000$ assets. The return of each asset follows a Merton's jump model [13]. We use this large data set to compare computational efficiency of GNCP and QBSVaR.

In order to generate the synthetic data set, we first want to generate a covariance matrix for the asset returns. Suppose the off-diagonal elements of the sample covariance matrix of data set DS1, Σ_o, have expected value d and standard deviation $\sqrt{v}$. The diagonal elements of Σ_o have expected value $\bar{d}$. We want to generate an $N \times N$ covariance matrix Σ such that it has the same expected value and standard deviation of the off-diagonal elements as Σ_o, where $N = 1000$. Hirschberger et al. [10] show that it is very difficult to generate a valid covariance matrix by randomly guessing such that the distributional characteristics (i.e., expected return and standard deviation) of the diagonal and off-diagonal elements are specified. They propose a procedure to generate such a covariance matrix.

Let l_{ij}, $i = 1, 2, \ldots, N$, $j = 1, 2, \ldots, \hat{m}$, be independent random variables identically distributed with some probability distribution that has mean $\hat{d}$ and standard deviation $\hat{v}$. Then the off-diagonal elements σ_{ij}, $i, j = 1, 2, \ldots, N$, $i \neq j$, in LL^T have expected value $d \geq 0$, variance $v \geq 0$ and the diagonal elements σ_{ii}, $i = 1, 2, \ldots, N$ have expected value $\bar{d} \geq 0$ if and only if

$$\hat{d} = \sqrt{\frac{d}{\hat{m}}} \tag{6.4}$$

$$\hat{v} = -\hat{d}^2 + \sqrt{\hat{d}^4 + \frac{v}{\hat{m}}} \tag{6.5}$$

$$\hat{m} = \frac{\bar{d}^2 - d^2}{v} \tag{6.6}$$

(see Hirschberger et al. [10], Theorem 1 and 2). We follow the procedure in [10], which is described in Algorithm 6.1, to generate the covariance matrix, Σ, with size 1000×1000 for the synthetic data set DS2.

After we generate a covariance matrix, we use the jump diffusion model [13] to generate the synthetic returns. A single asset price that has jumps and can be expressed as follows:

$$\frac{\mathrm{d}S_t}{S_t} = (\mu - k\lambda_J)\mathrm{d}t + \sigma \mathrm{d}W_t + (J - 1)\mathrm{d}\Pi_t \tag{6.7}$$

where μ is the mean return of the asset, W_t is the Wiener process, σ^2 is the variance of the asset, $\lambda_J > 0$ is the jump intensity, J is a random variable of jump amplitude with $\log J$

Algorithm 6.1: Generating Covariance Matrix with Specified Distributional Characteristics

```
Input:  $N$: the size of the covariance matrix to be generated;
$d$: the expected value of the off-diagonal elements in the covariance matrix;
$\sqrt{v}$: the standard deviation of the off-diagonal elements in the covariance matrix;
$\bar{d}$: the expected value of the diagonal elements in the covariance matrix.
Output: A covariance matrix $\Sigma_{N\times N}$ such that the expected value of the off-diagonal
        elements is $e$ and the standard deviation of the off-diagonal elements is $v$.
begin
    $\hat{m} \longleftarrow \frac{\bar{d}^2 - d^2}{v}$ ;
    Round $\hat{m}$ up or down to the nearest positive integer greater than 2 ;
    $\hat{d} \longleftarrow \sqrt{\frac{d}{\hat{m}}}$ ;
    $\hat{v} \longleftarrow -\hat{d}^2 + \sqrt{\hat{d}^4 + \frac{v}{\hat{m}}}$ ;
    Generate a matrix $Q_{N\times\hat{m}}$, each element $q_{ij}$ is independent and follows a standard
    normal distribution ;
    Generate a matrix $L_{N\times\hat{m}}$, such that $l_{ij} = \hat{d} + \sqrt{\hat{v}}q_{ij}$, $i = 1, 2, \ldots, N$,
    $j = 1, 2, \ldots, \hat{m}$ ;
    $\Sigma \longleftarrow LL^T$ ;
    return $\Sigma$ ;
end
```

normally distributed with a constant mean μ_J and a constant variance σ_J^2, $k = \mathrm{E}[J-1]$, Π_t is a Poisson counting process with

$$\mathrm{d}\Pi_t = \begin{cases} 1 & \text{with probability } \lambda_J \mathrm{d}t \\ 0 & \text{with probability } 1 - \lambda_J \mathrm{d}t \end{cases}. \tag{6.8}$$

For our synthetic data, we need to generate M returns for N asset prices. Let $\mathbf{S}_t \in \mathbb{R}^N$ be a vector representing N asset prices. We rewrite equation (6.7) as the following vector equation:

$$\frac{\mathrm{d}\mathbf{S}_t}{\mathbf{S}_t} = (\mu - \mathbf{k}\lambda_{\mathbf{J}})\mathrm{d}t + L \cdot \mathrm{d}\mathbf{W}_t + (\mathbf{J} - 1)\mathrm{d}\mathbf{\Pi}_t. \tag{6.9}$$

All the vector multiplication and division in equation (6.9) are element-wise multiplication and division. The vector μ is the mean return for N assets. L is an $N \times N$ matrix such that $LL^T = \Sigma$. $\mathbf{J}$ is a random vector of jump amplitude and each element of it is log normally distributed.

To generate random instance portfolio optimization problem, we randomly generate the mean return μ and the covariance matrix Σ. Each element of the mean return vector μ is independently drawn from a normal distribution with the mean and standard deviation equal to the mean and standard deviation of the mean returns of the assets in DS1. The way to generate Σ is specified above. We also set different jump parameters for each asset. μ_J is a number drawing from a uniform distribution in the interval $[-1.12, -0.72]$. σ_J is a number drawing from a uniform distribution in the interval $[0.325, 0.525]$. λ_J is a number drawing from a uniform distribution in the interval $[0.1, 0.2]$.

For each asset, we simulate $M = 100000$ prices using equation (6.9) and compute the returns using formula $ret = \frac{S_T - S_0}{S_0}$, where S_T is the underlying price after a time period $T = 1$ year and S_0 is the initial price. Figure 6.2 shows a return distribution of an asset in DS2, we can see that there is a bump on the left of the return distribution, representing a significant probability of unusually large losses.

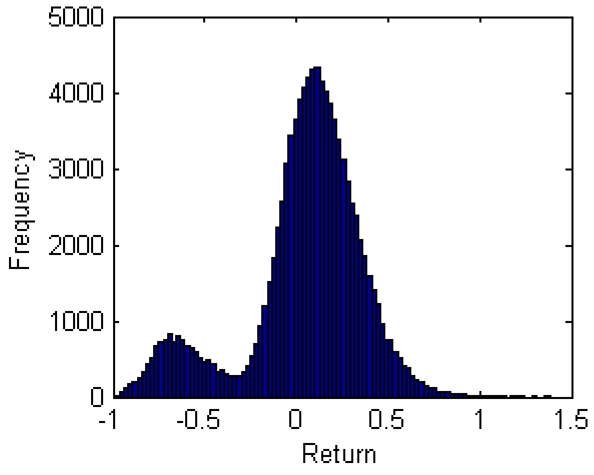

Figure 6.2: Return distribution of an asset in DS2

.

6.2.2 Comparing Computed VaR

In this section, we compare our gradual non-convexation penalty (GNCP) method with the quantile-based smoothed VaR (QBSVaR) method [9] in terms of computed VaR on the synthetic data set DS2. Similar to section 6.1.3, we compute the VaR of the minimal VaR portfolio for different number of assets, n, and different number of returns, m. We also compute the relative difference (6.3) of the VaR obtained using our GNCP method compared to that obtained using the QBSVaR method.

Table 6.5 shows the VaR obtained using the GNCP method and the relative difference with $\beta = 95\%$. We choose $\epsilon = 0.001$ for the smooth parameter of the QBSVaR method [9]. We do not report the relative difference when $m = 30000$ or $m = 100000$ since it takes more than 3 days to obtain the VaR optimal portfolio using the QBSVaR method. We will discuss the comparison for CPU time in section 6.2.3. We only report the cases when m is greater than n since this is usually the case in practice. We also report the computational

m	$n = 30$		$n = 50$		$n = 100$	
	VaR_{GNCP}	RD^Q_{VaR}	VaR_{GNCP}	RD^Q_{VaR}	VaR_{GNCP}	RD^Q_{VaR}
300	0.066506	10.48%	0.059168	-29.25%	0.020184	33.05%
1000	0.086351	4.16%	0.070866	0.68%	0.045326	16.14%
3000	0.094539	-0.36%	0.076082	6.29%	0.056926	16.04%
10000	0.094991	2.77%	0.081098	3.93%	0.069305	4.01%
30000	0.098859	-	0.085857	-	0.069741	-
100000	0.099429	-	0.084841	-	0.073519	-

Table 6.5: VaR of minimizing VaR portfolio using GNCP method and the relative difference compared to the QBSVaR method on synthetic data, $\beta = 95\%$, $\epsilon = 0.001$

m	n	VaR_{GNCP}	RD^Q_{VaR}		
			$\epsilon = 0.01$	$\epsilon = 0.001$	$\epsilon = 0.0001$
1000	500	0.014584	91.15%	198.20 %	220.87%
2000	500	0.026121	51.27%	82.58%	107.37%
2000	1000	0.034729	55.02%	16.37%	55.43%
5000	500	0.033475	90.65%	56.36%	85.08%
5000	1000	0.040264	-	24.30%	18.71%

Table 6.6: VaR of minimizing VaR portfolio using GNCP method and the relative difference compared to the QBSVaR method on synthetic data when n is large, $\beta = 95\%$

results when n is large in Table 6.6. The negative relative difference in Table 6.5 means the VaR obtained using the QBSVaR method [9] is smaller than that obtained using our GNCP method. However, this only occurs for two cases and one of them is very close to zero (-0.36%). For the case that the relative difference is positive, the relative difference is much more significant when n is large (most of the relative differences are more than 50% when $n = 500$ or $n = 1000$ in Table 6.6) for the synthetic data set DS2 compared to that for the historical data set DS1. This may be due to the fact that DS2 deviates further from normal distribution. Moreover, the test results in Table 6.6 also show that the choice of the smooth parameter, ϵ, of the QBSVaR method depends on the data set. For the synthetic data set DS2, the VaR corresponding to $\epsilon = 0.001$ is usually smaller than that corresponding to $\epsilon = 0.0001$ while for the historical data set DS1, this is not the case.

6.2.3 Comparisons in Computational Efficiency

In this section, we compare CPU time for our GNCP method and the QBSVaR method [9]. We measure the CPU time to compute optimal VaR portfolio for different number of assets, n, and different number of returns, m. We also measure the CPU time to evaluate the smoothed probabilistic constraint $H_\lambda(\mathbf{x}; \rho)$ of our GNCP method and the QBSVaR function $G_\epsilon(k, \mathbf{x})$ of the QBSVaR method [9] for different number of returns, m. Data set DS2 is used for this test since we need to see the running time of both methods as the problem size becomes large. We implement the two methods using Matlab. This may not be a proper programming language to measure the CPU time of the algorithm precisely since Matlab can do vector operations very efficiently. However, it is enough to compare the running time of two methods.

Figure 6.3 shows the CPU time (in seconds) to evaluate the smoothed probabilistic constraint $H_\lambda(\mathbf{x}; \rho)$ of our GNCP method and the QBSVaR function $G_\epsilon(k, \mathbf{x})$ of the QBSVaR method [9] for different number of returns, m, when $n = 100$ and $\beta = 95\%$. We fix $\mathbf{x} = \frac{1}{n} \times ones(n, 1)$ for both functions. For the smoothed probabilistic constraint, we choose

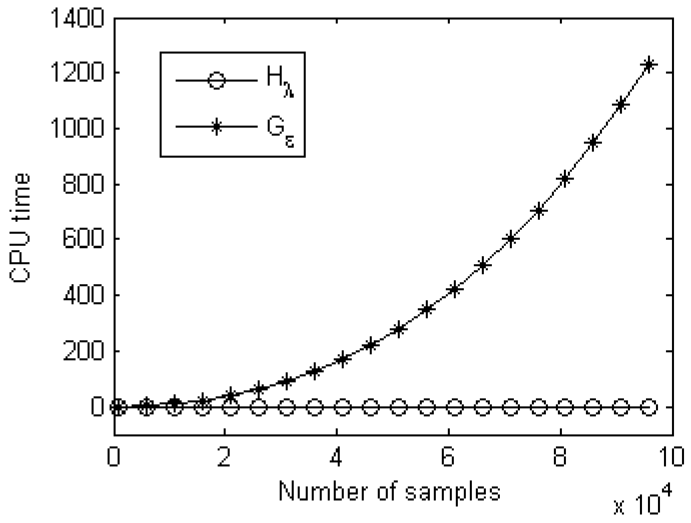

Figure 6.3: CPU time for function evaluation, $n = 100$, $\beta = 95\%$, $\epsilon = 0.001$

.

$\alpha = 0.01$, $\lambda = 4 \times 10^6$, and $\rho = 100$. For the QBSVaR function, we choose $\epsilon = 0.001$.

The star in Figure 6.3 represents the CPU time to evaluate $G_\epsilon(k, \mathbf{x})$ while the circle represents the CPU time to evaluate $H_\lambda(\mathbf{x}; \rho)$. We can see that the CPU time to evaluate $H_\lambda(\mathbf{x}; \rho)$ grows very slowly with respect to the number of samples, m. It is linear (with very small slope) with respect to m. However, the CPU time to evaluate $G_\epsilon(k, \mathbf{x})$ grows much faster than that to evaluate $H_\lambda(\mathbf{x}; \rho)$. More precise analysis shows that the CPU time is close to a cubic with respect to m. Figure 6.3 illustrates another main advantage of our GNCP method: it takes significantly less time to evaluate the objective and constraint functions and therefore, this method is computationally more efficient.

Table 6.7 reports the CPU time (in minutes) for both methods on the synthetic data set DS2 with $\beta = 95\%$. For the QBSVaR method [9], we choose $\epsilon = 0.01$ for the smooth parameter. We do not report the CPU time for QBSVaR method when $m = 30000$ or

	CPU Time (Minutes)								
m	$n = 30$			$n = 50$			$n = 100$		
	GNCP	QBSVaR	RT	GNCP	QBSVaR	RT	GNCP	QBSVaR	RT
300	0.19	1.83	9.64	0.18	3.05	16.76	0.30	6.22	20.72
1000	0.22	9.41	42.99	0.32	15.83	49.07	0.66	32.34	49.19
3000	0.53	56.04	104.92	0.76	95.70	126.67	1.40	190.10	135.37
10000	1.14	506.38	442.84	1.93	865.88	448.95	3.35	1789.40	534.26
30000	3.05	-	-	4.94	-	-	11.20	-	-
100000	9.82	-	-	16.54	-	-	34.41	-	-

Table 6.7: CPU time of running GNCP method and QBSVaR method on synthetic data and relative time, $\beta = 95\%$, $\epsilon = 0.001$

$m = 100000$ since it takes more than 3 days to obtain the VaR optimal portfolio using QBSVaR method. We also report the CPU time when n is large in Table C.2 in Appendix C. We compute the relative time (RT), which is defined in equation (6.10), to provide a more clear comparison between the two methods.

$$RT = \frac{Time_{QBSVaR}}{Time_{GNCP}} \tag{6.10}$$

We make following observations based on Table 6.7 and Table C.2.

1. For the QBSVaR method [9], the smaller the smooth parameter, ϵ, is, the less time it takes to solve the VaR minimization problem (2.8). According to equation (4.15), when ϵ is small, the number of losses that satisfies the constraint $|G(k, \mathbf{x}) - f_i(\mathbf{x})| \leq \epsilon$ is small and this reduces the time to evaluate the QBSVaR function $G_\epsilon(k, \mathbf{x})$.

2. Our GNCP method takes less CPU time to solve the VaR minimization problem (2.8) than the QBSVaR method [9]. The relative time RT is more than 40 when $m \geq 1000$ and n is small. This means our GNCP method solves the VaR minimization problem (2.8) over 40 times faster than the QBSVaR method [9].

3. For a fixed number of assets, n, the CPU time of our GNCP method grows much slower than the speed of m increases, i.e., the CPU time is less than linear with respect to m. However, the CPU time of the QBSVaR method [9] grows much faster (faster than quadratic) with respect to m. This observation demonstrates the most significant merit of our GNCP method: the time for function and constraint evaluation is much less compared to the QBSVaR method.

4. For a fixed number of returns, m, the CPU time of the QBSVaR method [9] grows linearly with respect to n. When n is small, the CPU time of our GNCP method also grows linearly with respect to n. However, when n becomes large, the CPU time of our GNCP method grows a little faster. The advantage of our GNCP method is that the time for function evaluation is linear with respect to m. When m is small and n is large, the time for solving the linear system in the optimization procedure dominates the total running time and therefore the advantage of our GNCP method is less prominent. However, in practice, the number of returns , m, is usually much larger than the number of assets, n. Taking both m and n into consideration, the CPU time of the QBSVaR method grows faster as the problem size becomes larger.

To conclude, the choice of the smooth parameter, ϵ, in the QBSVaR method [9] not only determines how accurate the optimal solution is, but also has an impact on the running time of the method. Our GNCP method is much more efficient than the QBSVaR method [9] in terms of running time, especially when the number of returns, m, is large.

6.3 Efficient Frontiers

In practice, in addition to minimum VaR portfolio, we want to consider a VaR frontier. For different levels of expected return target R, we can solve problem (2.7) and obtain the corresponding VaR of the optimal portfolios. Then we draw the plot of the (VaR, Return) pair in the VaR-Return plane. This gives us an efficient frontier.

We determine the range of the possible return target $[R_{min}, R_{max}]$ as follows. We solve the maximum return optimization problem (6.11) to get the maximum possible return target R_{max}. Then we solve the VaR minimization problem (2.8) and set the corresponding expected return as the minimum possible return target R_{min}.

$$\begin{aligned} \max_{\mathbf{x}} \quad & \mathrm{E}(\mathbf{x}^T\mathbf{r}) \\ \text{s.t.} \quad & \sum_{i=1}^{n} x_i = 1 \\ & x_i \geq 0 \qquad i = 1, 2, \ldots, n \end{aligned} \tag{6.11}$$

Figure 6.4 shows the VaR-Return efficient frontier of the samples for the historical data set DS1 with $m = 500$, $n = 10$ and $\beta = 95\%$. Figure 6.5 shows the VaR-Return efficient frontier of the samples for the synthetic data set DS2 with $m = 500$, $n = 10$ and $\beta = 95\%$. The solid line is the efficient frontier we obtain by using our GNCP method. The dotted line is the efficient frontier we obtain by using the QBSVaR method [9]. We also tested different number of assets and different number of returns. The results are similar.

From Figure 6.4 and Figure 6.5, we can see that the efficient frontier corresponding to our GNCP method dominates that corresponding to the QBSVaR method [9]. For a fixed return R, the VaR we obtain using our GNCP method is smaller than that we obtain using the QBSVaR method [9]. This proves that our GNDP method not only yields better VaR portfolio than the QBSVaR method [9] for problem (2.8), but also yields better VaR

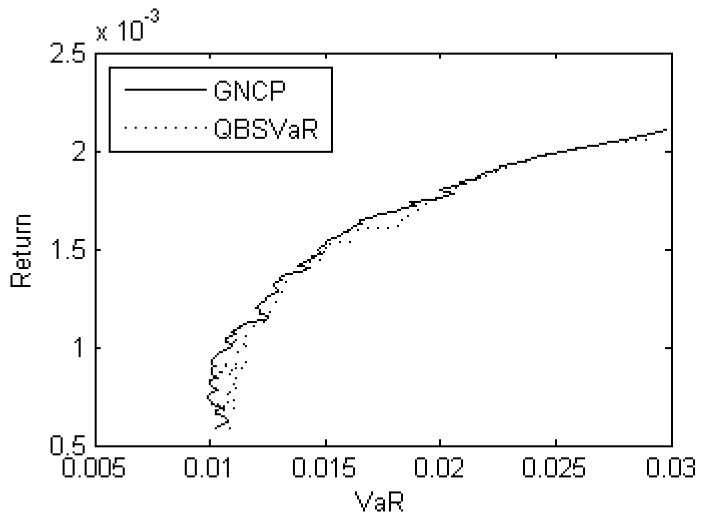

Figure 6.4: Efficient frontier in the VaR-Return plane on historical data, $m = 500$, $n = 10$, $\beta = 95\%$

.

portfolio for problem (2.7). We can also see that the efficient frontier obtained using the GNCP method is better at the left end point than the right end point especially for the historical data in Figure 6.4. This is reasonable since the right end point corresponds to the maximum return portfolio. However, for the synthetic data in Figure 6.5, the VaR obtained using our GNCP method is also better than that obtained using QBSVaR method when the expected return target is very large.

The efficient frontier for Return vs VaR is not very smooth in Figure 6.4 and Figure 6.5. Some of the points are not Pareto optimal. Similar result can also be found in Gaivoronski and Pflug [9]. The reason for this is that the optimal solution we get for problem (2.7) is an approximation solution. Figure 6.6 shows the VaR-Return efficient frontier of the samples for the historical data set DS1 with $m = 1000$, $n = 20$ and $\beta = 95\%$. We see that the efficient frontier will be smoother as the number of samples increases.

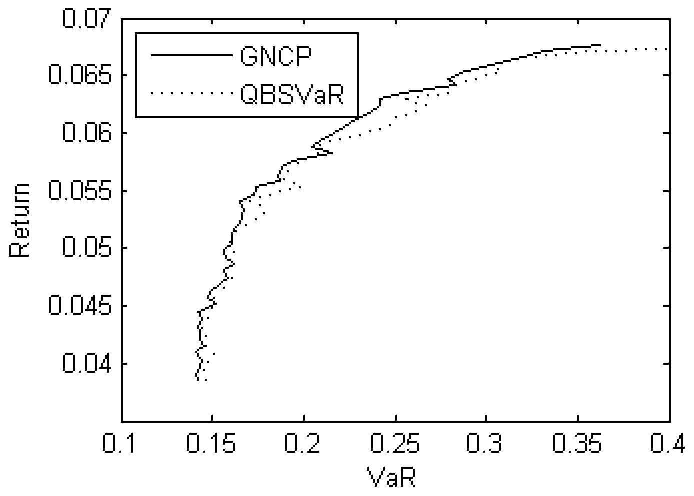

Figure 6.5: Efficient frontier in the VaR-Return plane on synthetic data, $m = 500$, $n = 10$, $\beta = 95\%$

.

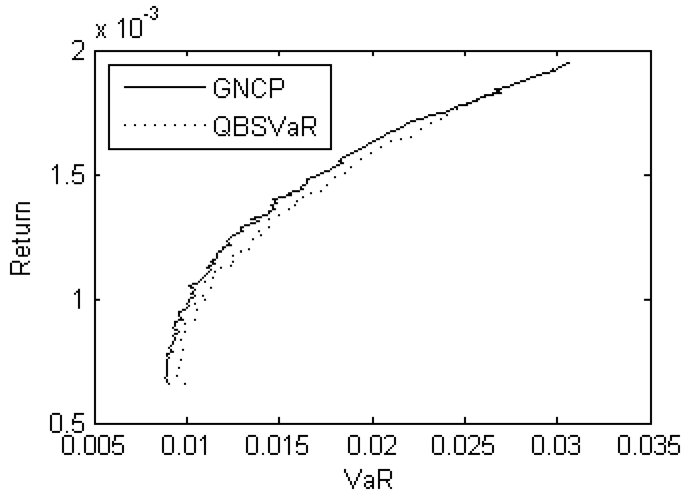

Figure 6.6: Efficient frontier in the VaR-Return plane on historical data, $m = 1000$, $n = 20$, $\beta = 95\%$

.

Chapter 7

Conclusions

7.1 Conclusion

The VaR minimization problem is computationally difficult to solve. Given discrete samples, VaR function is non-convex, non-smooth, and has many local minima. The CVaR minimization method [19] is an efficient method to solve CVaR minimization problem. However, a minimal CVaR portfolio does not necessarily have a minimal VaR, even though the VaR of a portfolio is always smaller than the CVaR of a portfolio. We confirm that our GNCP VaR minimization method always yields smaller VaR than the CVaR minimization method for the same VaR minimization problem on a small historical data set.

Gaivoronski and Pflug [9] propose a quantile-based smoothed VaR (QBSVaR) method that minimizes a quantile-baed smoothed VaR function, which is an approximation of the VaR function. They also provide the evidence that their QBSVaR method is better than the CVaR minimization method [19] when being used to obtain an optimal VaR portfolio. Their approach is based on the definition that VaR is a quantile of the portfolio losses. They reformulate the quantile as a linear combination of the loss functions with the coefficients being some indicator functions that can be written as the product of step functions. They approximate the quantile by replacing the step function with a smooth approximation function. Their approach is complex since they need to enumerate all possible combinations of choosing k indices from m numbers. A smooth parameter, ϵ, is introduced to control how accurate the approximation is. Usually a smaller value of ϵ makes the approximation more accurate but the the optimization problem has more local minima and is more difficult to solve. On the other hand, a larger value of ϵ makes the approximation crude while the optimization problem has less local minima. The difficulty in choosing the smooth parameter, ϵ, is one of the main problem of this smooth method. Another drawback of this smooth method is the computational complexity. This method takes $O(m^3)$ time to evaluate the objective function of the optimization problem in addition to evaluating each loss function of the portfolio where m is the number of samples. In addition, optimization methods typically require gradient and hessian evaluations. This makes it practically impossible to solve a problem when the number of samples is large.

We propose a gradual non-convexation penalty (GNCP) method for the VaR minimization problem. We introduce an auxiliary variable α and define VaR_β as the minimum α value which satisfies the constraint that the probability of loss greater than α cannot exceed $1 - \beta$. The probabilistic constraint is written as a sum of step indicator functions. Similar to the QBSVaR method [9], we use a piecewise quadratic function to approximate the step indicator function. The approximate optimization problem is solved by using an exact penalty method. In addition, we use a gradual non-convexation process in an attempt to reach a global minimizer. This process is indexed by a parameter ρ, which controls how accurate the approximation is. When the parameter, ρ, is small, the approximation is close to a convex problem and the optimization problem is easy to solve. It captures the global characteristic of the function. As the parameter goes to infinity, the approximation approaches to the original VaR minimization problem while the approximation itself becomes difficult to solve. We gradually increase the parameter and solve the original VaR minimization problem by solving a sequence of nonlinearly constrained optimization problems using an exact penalty method. This does not significantly increase the total computational time since the optimal solution for each step is a good initial point for the next one and it typically takes a small number of optimization iterations to solve the optimization problem for each indexing parameter. We also show that our new method takes only $O(m)$ extra time to evaluate the objective function of the optimization problem This is a drastic improvement compared to the QBSVaR method [9], which takes $O(m^3)$ time for function evaluation.

Both historical and synthetic data are used to evaluate performance of our GNCP method and the QBSVaR method [9]. Our computational results demonstrate that the GNCP method yields significantly better VaR for the VaR minimization problem than the QBSVaR method [9] most of the time. The computed results for our GNCP method are 5$\sim$10% better on average for historical data and even better for synthetic data when the number of assets, n, is large. The efficient frontier of our GNCP method dominates that of the QBSVaR method. We also show that it is hard to choose a smooth parameter, ϵ, for the QBSVaR method. Tests on large size problems numerically show that our GNCP method is much more computationally efficient than the QBSVaR method [9] especially when the number of samples, m is large. The CPU time for our GNCP method is less than linear with respect to m while that for the QBSVaR method is more than quadratic with respect to m.

7.2 Possible Future Work

In this thesis, we use a continuous and differentiable function $g_\lambda(z;\rho)$ to replace the continuous function $h_\lambda(z)$ for the step function $\mathbb{I}_+(z)$. However, this function is not twice continuously differentiable. For an optimization solver, the problem may be solved faster and the optimal solution may be more accurate if the objective function or the constraint function is twice continuously differentiable. A function similar to the approximation function $\varphi_\epsilon(z)$ in the QBSVaR method [9] may be a good alternative to $g_\lambda(z;\rho)$.

We have shown that the gradual non-convexatoin penalty method can be used to solve the optimization problem with a probability constraint very efficiently. We can apply this technique to other challenging optimization problems with a probability constraint.

APPENDICES

Appendix A

Stocks in Data Set 1

1	AA	11	HD	21	MMM
2	AXP	12	HON	22	MO
3	BA	13	HWP	23	MRK
4	C	14	IBM	24	MSFT
5	CAT	15	INTC	25	PG
6	DD	16	IP	26	SBC
7	DIS	17	JNJ	27	T
8	EK	18	JPM	28	UTX
9	GE	19	KO	29	WMT
10	GM	20	MCD	30	XOM

Table A.1: Stocks in data set 1

Appendix B

More Results on Suboptimality of VaR from CVaR Minimization

m	n	VaR		RD^{C}_{VaR}
		GNCP	min CVaR	
300	10	0.0079865	0.0092844	16.25%
300	20	0.0072237	0.0088394	22.37%
300	30	0.0051924	0.0069731	34.29%
500	10	0.0078666	0.008292	5.41%
500	20	0.0072434	0.0081669	12.75%
500	30	0.0056612	0.006902	21.92%
1000	10	0.0076282	0.0085471	12.05%
1000	20	0.0071986	0.0074057	2.88%
1000	30	0.0056484	0.0065554	16.06%
2000	10	0.010026	0.010508	4.81%
2000	20	0.0086397	0.0091924	6.40%
2000	30	0.0075813	0.0081635	7.68%

Table B.1: VaR and relative difference of minimizing VaR and minimizing CVaR portfolio, $\beta = 90\%$

m	n	CVaR		RD_{CVaR}
		GNCP	min CVaR	
300	10	0.019276	0.017353	-1.44%
300	10	0.015253	0.014073	-7.74%
300	20	0.013687	0.012864	-6.01%
300	30	0.012526	0.010227	-18.35%
500	10	0.013545	0.013004	-3.99%
500	20	0.012991	0.01166	-10.25%
500	30	0.011573	0.0098052	-15.28%
1000	10	0.013552	0.012935	-4.55%
1000	20	0.011957	0.011421	-4.48%
1000	30	0.010473	0.0098365	-6.08%
2000	10	0.01777	0.016789	-5.52%
2000	20	0.015717	0.014987	-4.64%
2000	30	0.014327	0.013473	-5.96%

Table B.2: CVaR and relative difference of minimizing VaR and minimizing CVaR portfolio, $\beta = 90\%$

Appendix C

More Results on Comparison Between GNCP and QBSVaR

m	n	VaR_{GNCP}	RD^{Q}_{VaR}			
			$\epsilon = 0.01$	$\epsilon = 0.001$	$\epsilon = 0.0001$	$\epsilon = 0.00001$
300	10	0.0079865	17.45%	19.39%	4.17%	8.56%
300	20	0.0072237	18.54%	9.79%	3.92%	11.94%
300	30	0.0051924	54.74%	28.17%	14.92%	29.28%
500	10	0.0078666	11.05%	3.41%	2.08%	2.49%
500	20	0.0072434	13.15%	-1.98%	5.16%	6.59%
500	30	0.0056612	25.58%	9.30%	7.53%	4.64%
1000	10	0.0076282	10.54%	5.53%	3.52%	8.84%
1000	20	0.0071986	2.45%	1.29%	1.76%	0.45%
1000	30	0.0056484	16.77%	10.68%	10.55%	10.20%
2000	10	0.010026	6.81%	3.20%	4.81%	3.32%
2000	20	0.0086397	6.22%	2.11%	6.28%	7.44%
2000	30	0.0075813	9.50%	9.91%	9.72%	8.96%

Table C.1: VaR of minimizing VaR portfolio using GNCP method and the relative difference compared to the QBSVaR method on historical data, $\beta = 90\%$

m	n	CPU Time (Minutes)				RT, $\epsilon = 0.001$
		GNCP	QBSVaR			
			$\epsilon = 0.01$	$\epsilon = 0.001$	$\epsilon = 0.0001$	
1000	500	8.41	293.23	163.53	140.34	19.46
2000	500	10.25	1225.59	483.75	418.31	47.18
2000	1000	51.70	2369.37	986.47	847.74	19.08
5000	500	17.04	11218.49	2396.22	2086.44	140.61
5000	1000	73.61	-	4217.67	4854.85	57.30

Table C.2: CPU time of running GNCP method and QBSVaR method on synthetic data and relative time when n is large, $\beta = 95\%$

References

[1] S. Alexander, T. F. Coleman, and Yuying Li. Minimizing var and cvar for a portfolio of derivatives. *Journal of Banking and Finance*, 30(2):583–605, 2006.

[2] Ph. Artzner, F. Delbaen, J.-M. Eber, and D.Heath. Coherent measures of risk. *Mathematical Finance*, 9(3):203–228, 1999.

[3] Fischer Black and Myron Scholes. The pricing of options and corporate liabilities. *Journal of Political Economy*, 81(3):637–654, 1973.

[4] Stephen Boyd and Lieven Vandenberghe. *Convex Optimizaion*. Cambridge University Press, 2004.

[5] T. F. Coleman, J. Henninger, and Y. Li. Minimizing tracking error while restricting the number of assets. *Journal of Risk*, 8:33–56, 2006.

[6] M. Crouhy, D. Galai, and R. Mark. *Risk Management*. McGraw-Hill, 2001.

[7] Kevin Down. *Measuring Market Risk*. John Wiley & Sons, Ltd, second edition, 2005.

[8] D. Duffie and J. Pan. An overview of value at risk. *Journal of Derivatives*, 4:9–49, Spring 1997.

[9] Alexei A. Gaivoronski and Georg Pflug. Value-at-risk in portfolio ptimization: Properties and computational approach. *Journal of Risk*, 7(2):1–31, Winter 2004-2005.

[10] Markus Hirschberger, Yue Qi, and Ralph E. Steuer. Randomly generating portfolio-selection covariance matrices with specified distributional characteristics. *European Journal of Operational Research*, 177:1610–1625, 2007.

[11] N. Larsen, H.Mausser, and S. Uryasev. Algorithm for optimization of value-at-risk. In P. Pardalos and V.K. Tsitsiringos, editors, *Financial Engineering, e-Commerce and Supply Chain*, pages 129–157. Kluwer Academic Publishers, 2002.

[12] H.M. Markowitz. Portfolio selection. *Journal of Finance*, 7(1):77–91, 1952.

[13] Robert C. Merton. Option pricing when underlying stock returns are discontinuous. *Journal of Financial Economics*, 3:125–144, 1976.

[14] Basle Committee on Banking Supervision. Amendment to the capital accord to incorporate market risks, 1996.

[15] A. Pfingsten, P.Wagner, and C. Wolferink. An empirical investigation of the rank correlation between different risk measures. *Journal of Risk*, 6(4):55–74, 2004.

[16] Georg Pflug. Some remarks on the value-at-risk and the conditional value-at-risk. In S. Uryasev, editor, *Probabilistic Constrained Optimization.* Kluwer Academic Publishers, 2000.

[17] Sergey Sarykalin, Gaia Serraino, and Stan Uryasev. Value-at-risk vs. conditional value-at-risk in risk management and optimization. *Tutorials in Operations Research*, pages 270–294, 2008.

[18] William F. Sharpe. Capital asset prices: A theory of market equilibrium under conditions of risk. *Journal of Finance*, 19(3):425–442, 1964.

[19] S. Uryasev and R.T. Rockafellar. Optimization of conditional value-at-risk. Technical Report 99-4, Center for Applied Optimization at the University of Florida, 1999.

[20] David Wozabal. A new method for value-at-risk constrained optimization using the difference of convex algorithm. *CARIPLO Workshop*, September 2008.