Chance Constrained Programs with Gaussian Mixture Models

Zhaolin Hu∗ Wenjie Sun† Shushang Zhu‡

Abstract

In this paper, we discuss input modeling and solution techniques for several classes of chance constrained programs (CCPs). We propose to use Gaussian mixture models (GMM), a semi-parametric approach, to fit the data available and to model the randomness. We demonstrate the merits of using GMM. We consider several scenarios that arise from practical applications and analyze how the problem structures could embrace alternative optimization techniques. More specifically, for several scenarios, we study how to assess the gradient of the chance constraint and incorporate the results into gradient-based nonlinear optimization algorithms, and for a class of CCPs, we propose a spatial branch-and-bound (BB) procedure and solve the problems to global optimality. We also conduct numerical experiments to test the efficiency of our approach and propose an example of hedge fund portfolio to illustrate the practical application of the method.

Keywords: Stochastic Programming, Chance Constrained Programs, Gaussian Mixture Model, Spatial Branch-and-Bound, Value-at-Risk

1 Introduction

Stochastic optimization (SO) has become a standard modeling philosophy and framework for decision making under uncertainty. Chance constrained programs (CCP), as a basic class of SO models, have been used in various areas. When there are random parameters in the constraints of an optimization problem, it is natural to require the constraints be satisfied with a given high probability and formulate the problem as a CCP. The literature on CCPs can be dated back to Charnes et al. (1958), who first considered a single CCP (SCCP), and Miller and Wagner (1965), who first considered a joint CCP (JCCP). Since then, both theories and applications of CCPs have been studied extensively. For a comprehensive study and literature review on the topic, readers are referred to Prékopa (2003), Nemirovski and Shapiro (2006), Hong et al. (2011), Zymler et al. (2013), Hanasusanto et al. (2017), Geng and Xie (2019) and references therein.

∗School of Economics and Management, Tongji University, Shanghai 200092, China. Email: russel-l@tongji.edu.cn.
†Department of Analytics and Operations, NUS Business School, National University of Singapore, 119245, Singapore. Email: wenjie_sun@u.nus.edu.
‡Corresponding author. Department of Finance and Investment, Business School, Sun Yat-Sen University, Guangzhou 510275, China. Email: zhuss@mail.sysu.edu.cn.
In the framework of SO, the randomness of the system is modeled by random parameters. The process of specifying a distribution for the random parameters is often called input modeling. Input modeling may directly affect the optimization results and thus affect the decision. In practice, there is usually no distribution that is ready for use, and decision makers need to conduct input modeling based on the data/information available. In this paper, we make the following emphasis: Since the optimal solutions of CCPs critically depend on the input distributions, input modeling is a critical module for CCPs, and is perhaps equally important as the solution techniques. We conduct an integrated study between the input modeling module and the optimization module.

Input modeling has more deeply been studied in the area of simulation, see, e.g., Law (2013). When a simulation model is being built, the modeler wants to specify the input distribution for the random parameters that drive the stochastic system. There have been various approaches for input modeling in simulation. When there are data, one can fit a distribution to the data. When there are no data, the modeler often needs to specify the distribution based on some subjective judgment. In simulation, the data based approaches can roughly be divided into the parametric approach and the nonparametric approach. In the parametric approach, decision makers select a parametric distribution family, e.g., a multivariate normal (Gaussian) distribution, and then use statistical methods to determine (estimate) the parameters of the distribution, e.g., the mean vector and the covariance matrix. In the nonparametric approach, decision makers do not specify a parametric family. Rather, they use some nonparametric methods to fit the distribution. For instance, the kernel density estimation methods, which can be dated back to Rosenblatt (1956) and Parzen (1962), are often used in practice.

Nowadays we have witnessed an era of data. In many real SO problems, we can obtain data for the random parameters. Thus it is often beneficial to mining the data to specify the input distribution for SO. This kind of data driven approach has been very natural and attractive. In this paper we propose such an input modeling approach for CCPs. The interface between input modeling and CCPs was considered in the literature. For instance, Henrion and Möller (2012) and Bremer et al. (2015) studied CCPs for which they modeled the underlying distribution as normal (or normal-like) distributions. In this paper, we propose to use mixture distributions (or called mixture models) to model the randomness of the parameters. Note that a mixture distribution is a weighted sum of a finite set of probability measures. In contrast to the parametric approach and the nonparametric approach, the use of mixture distribution can be viewed as a semiparametric approach (McLachlan and Peel 2000, Bishop 2006). The idea of using mixture distribution can be dated back to Pearson (1894), who proposed to use a mixture of two normal distributions to fit a set of asymmetric biological data. Pearson’s approach led to new findings in the biological area. Mixture models allow for sufficient flexibility (see, e.g., McLachlan and Peel (2000) and Frühwirth-Schnatter (2006)). They provide a framework of constructing more complex distributions. For instance, the asymmetrical distributions and skewed distributions can often be approximated well by the mixture models (e.g., Wang and Taaffe 2015). Among various mixture distributions, we are particularly interested in the Gaussian mixture model (GMM), which has been studied extensively in the literature, and has been
GMM has been used in the distributionally robust CCPs recently. For instance, Chen et al. (2018) considered a GMM based distributionally robust CCP, where they assumed the weights of the GMM belong to some uncertainty set. They proposed lower and upper approximations to their model under certain conditions. Lasserre and Weisser (2021) investigated a more general mixture setting. They built inner approximation sets constructed by polynomials to the feasible region of their distributionally robust CCP. Instead of exploring robust optimization, our work studies conventional CCPs. We use GMM to conduct input modeling (to learn the underlying distribution), and propose optimization procedures (gradient based method and spatial branch-and-bound algorithm) to directly solve different classes of CCPs under the learned GMM. Bringing in the mixture input models in CCPs, we show that our approach has many merits and can extend the current research results for CCPs in some aspect.

It is well known that CCPs are in general difficult to solve. The modelers often need to consider special structures of the CCPs for handling them. In this paper, we consider several scenarios that arise from real applications and discuss how the chance constraints can be simplified accordingly. As a straightforward but important result, we show that the probability function in a CCP with a mixture distribution can be transformed as a weighted sum of probability functions with component distributions of the mixture distribution. Based on the result, we analyze a class of JCCPs and a class of SCCPs.

For JCCPs, we consider a class of CCPs in which the constraints within the probability function have a linear structure, and the random vector and the decision vector are separated. Many engineering management problems can be modeled as such CCPs. The electricity network capacity optimization problem studied by Henrion and Möller (2012) and the renewable energy management problem studied by Bremer et al. (2015) are typical examples. When the underlying distribution is a (possibly singular) Gaussian distribution, certain results about the gradient of the probability function have been derived. For instance, Henrion and Möller (2012) provided some explicit formula for the probability function which allows one to reduce the calculation of gradients to the calculation of probability function values. Nevertheless, it turns out to be difficult to build similar results for other (parametric) distribution families. In this paper, we show that the results could be easily generalized for the GMM. For SCCPs, we consider a class of CCPs where the constraint within the probability function takes a product form of the randomness and the decision vector. This class of models have found applications in various areas. We show that for the GMM, the problems can be reformulated as deterministic optimization problems where the standard normal distribution function plays a substantial role. Therefore, we can directly use conventional nonlinear optimization techniques to handle the problems. Note that the nonlinear optimization techniques may not guarantee the global optimality. In this paper, we further explore the problem structure and propose a globally convergent spatial branch-and-bound (BB) algorithm to solve this class of CCPs.

The main contributions of this paper are summarized as follows:

- We consider the integration of the input modeling module and the solution module when
using CCPs in practice, and highlight the importance of such integration. This integrated approach could benefit management practice.

- We propose to use GMM to model the random distribution in the CCP. GMM provides sufficient freedom to learn the data information.

- We explore structures and properties for a number of CCPs under the GMM framework, and develop efficient optimization methods to solve different classes of CCPs. More specifically, for several scenarios, we study how to assess the gradient of the chance constraint under GMM and incorporate the results into gradient-based nonlinear optimization algorithms, and for a class of CCPs, we propose a spatial branch-and-bound (BB) procedure and solve the problems to global optimality.

The rest of this paper is organized as follows. In Section 2 we introduce the mixture models and analyze the CCPs with the mixture models. In Section 3 we consider a number of scenarios and demonstrate the potential wide applications of our approach. We conduct numerical study in Section 4. Section 5 concludes the paper. Some supplementary materials are provided in the appendix.

2 Mixture Distributions and Model Structures

We start from the general chance constrained program (CCP) paradigm:

\[
\begin{align*}
\text{minimize} & \quad h(x) \\
\text{subject to} & \quad \Pr_{\sim P^*} \{c_1(x, \xi) \leq 0, \cdots, c_m(x, \xi) \leq 0\} \geq 1 - \alpha.
\end{align*}
\]

In Problem (1a)-(1b), \( x \) is the decision vector, which belongs to a set \( X \subset \mathbb{R}^d \), \( h : \mathbb{R}^d \rightarrow \mathbb{R} \) is a real-valued function which models the objective, \( \xi \) is a \( k \)-dimensional vector of random parameters and the support of \( \xi \), denoted as \( \Xi \), is a closed subset of \( \mathbb{R}^k \), \( c_i : \mathbb{R}^d \times \Xi \rightarrow \mathbb{R}, i = 1, \cdots, m \) are real-valued functions, and the notation \( \Pr_{\sim P^*} \) denotes that the probability is taken with respect to \( P^* \), where \( P^* \) is referred to as the underlying distribution. Constraint (1b) is called a chance constraint. It requires that the \( m \) constraints are satisfied at least with a probability \( 1 - \alpha \). Problem (1a)-(1b) is called a SCCP if \( m = 1 \) and a JCCP if \( m > 1 \). This is a very natural requirement in mathematical programs when random parameters are involved in the constraints. Throughout this paper we assume that \( X \) is a convex and compact set, which may be defined by some deterministic constraints, and the function \( h \) can be handled by deterministic optimization techniques. Thus, the major difficulty of solving Problem (1a)-(1b) comes from the chance constraint (1b).

While the idea of CCP is conceptually very natural and fits decision making very well, solving a CCP is often a nontrivial task. In general, CCPs are challenging problems. There are a number of reasons for this. For instance, the probability function in a CCP typically has no closed form, and the convexity of the probability function is not easy to preserve.
Many approaches have been developed to tackling CCPs. A natural approach is to treat the probability function as a nonlinear function and use nonlinear optimization techniques (e.g., Henrion and Möller (2012) and Bremer et al. (2015)). Besides this, two streams of approaches are widely used. One stream can be termed as a deterministic optimization approach. In this approach, some deterministic optimization model is constructed to approximate the CCP. The deterministic model is then solved and the solution is used to approximate that of the CCP. It includes the tractable safe approximations and the robust optimization framework, which have been explored extensively in the optimization literature, see, e.g., Nemirovski and Shapiro (2006) and Chen et al. (2010). Another stream is to use a sampling based approach. In the approach, the sample used are simulated from a specified distribution or are obtained from historical data. Given the sample, a sample problem is constructed associated with the CCP and different optimization procedures are then designed to solve the sample problem. These include the sample average approximation (SAA) approach (see, e.g., Luedtke and Ahmed (2008), Pagnoncelli et al. (2009) and Luedtke et al. (2010)), the scenario approach (Calafiore and Campi (2005, 2006)), the p-efficient point based approach (Dentcheva et al. 2000), and the binarization approach (Lejeune 2012). In the simulation framework, the CCP is often treated or transformed as a nonlinear optimization problem, and simulation based methods are developed to solve it, see, e.g., Hong et al. (2011), Henrion (2013), Hu et al. (2013) and Peña-Ordieres et al. (2019).

2.1 Probability Function with Mixture Distribution

When Problem (1a)-(1b) is being studied, the underlying distribution $P^*$ is an important module. However, as discussed in Section 1, in practice, such a distribution is usually not pre-given exactly and can only be inferred. In this paper, we model that $P^*$ has the following density

$$p^*(z) = \sum_{j=1}^{K} \pi_j p_j(z)$$

with the Lebesgue measure, where $p_j(z), j = 1, \cdots, K$ are probability densities which are called component distributions, and $\pi_j > 0, j = 1, \cdots, K$ are mixing coefficients satisfying $\sum_{j=1}^{K} \pi_j = 1$. The distribution with density (2) is called a mixture distribution (or mixture model), see, e.g., McLachlan and Peel (2000). Note that throughout the analysis we do not differentiate the notations $P^*$ and $p^*$ when there is no confusion. Both notations refer to the same distribution. Note also that in (2), $p^*$ and $p_j$ could be probability mass functions in which case a counting measure is adopted.
2.2 Gaussian Mixture Model

A most well known instance of the mixture models is the GMM. Following the notion in Bishop (2006), a GMM has the following density

\[ p_\ast(z) = \sum_{j=1}^{K} \pi_j \mathcal{N}(z | \mu_j, \Sigma_j), \]

where each component \( \mathcal{N}(z | \mu_j, \Sigma_j) \) is a (multivariate) normal density function with mean vector \( \mu_j \) and covariance matrix \( \Sigma_j \). Throughout the paper we assume that \( \Sigma_j, j = 1, \ldots, K \) are positive definite. The GMM is widely used in machine learning literature, see, e.g., Bishop (2006). Using GMM to approximate a distribution has been very popular in the literature, see, for instance, Zeevi and Meir (1997), Li and Barron (2000) and Maugis-Rabusseau and Michel (2013). Wilson and Calway (2001) built that the GMM can approximate a given distribution to any accuracy under some metric. Recently, GMMs have also been used to learn the response surface and to guide the random search in simulation optimization, see, e.g., Sun et al. (2018).

In practice, the distribution of the random parameters in a CCP may be skewed, asymmetrical, or multimodal. These distribution structures are typically hidden in the data that are collected. Consequently, conventional parametric families, e.g., a Gaussian distribution or an exponential family may not fit the data well. In such situations, it may be better to use some mixture distribution to model the randomness.

As indicated in the literature, there are several issues for the use of mixture models. One essential issue is how to fit the mixture models to the data. There are a number of approaches that have been developed for such fitting. A widely used method is the expectation maximization (EM) method, see, for instance, Bishop (2006). Suppose we have an independent and identically distributed (i.i.d.) sample from \( \xi \), denoted as \( \xi_1, \xi_2, \ldots, \xi_N \). We can use the following EM algorithm (Chapter 9 of Bishop 2006) to obtain a GMM.

**Expectation Maximization (EM)**

**Step 1.** Initialize \( \mu_j, \Sigma_j \) and \( \pi_j, j = 1, \ldots, K \).

**Step 2.** E Step. Compute

\[ \gamma_{nj} = \frac{\pi_j \mathcal{N}(\xi_n | \mu_j, \Sigma_j)}{\sum_{i=1}^{K} \pi_i \mathcal{N}(\xi_n | \mu_i, \Sigma_i)}. \]

**Step 3.** M Step. Re-estimate

\[ \mu_j^{\text{new}} = \frac{1}{N_j} \sum_{n=1}^{N} \gamma_{nj} \xi_n, \quad \Sigma_j^{\text{new}} = \frac{1}{N_j} \sum_{n=1}^{N} \gamma_{nj} (\xi_n - \mu_j^{\text{new}})(\xi_n - \mu_j^{\text{new}})^\top, \quad \pi_j^{\text{new}} = \frac{N_j}{N}, \]

where

\[ N_j = \sum_{n=1}^{N} \gamma_{nj}. \]
Step 4. Check whether the convergence criterion is satisfied. If not, return to Step 2.

The convergence of the EM algorithm has been built in the literature, see, e.g., Wu (1983). In Step 4, a convergence criterion should be specified. A widely used rule in the literature is to terminate the algorithm if the difference change of the likelihood (or log-likelihood) function is smaller than a given small value (e.g., McLachlan and Peel 2000). The EM algorithm has been embedded in some softwares. For instance, in Matlab, we can directly call the EM algorithm package to conduct the estimation for the GMM and set the termination rule accordingly.

2.3 Model Structures

The simulation approach is general and may be used to handle various problem structures and distributions. Since our mixture model can be viewed as a general distribution, the simulation approach may be directly used to handle the problem. However, the detailed procedures of this approach often significantly depend on the probability function structures and the underlying distribution.

In this paper, we explore structures and properties of constraint (1b) when the underlying distribution is a mixture distribution, and use them to design efficient procedures to solve Problem (1a)-(1b) under different scenarios. Let \( c(x, \xi) = \max \{ c_1(x, \xi), \ldots, c_m(x, \xi) \} \). Then the constraint (1b) can be rewritten as \( \Pr_{\sim P_{\ast}} \{ c(x, \xi) \leq 0 \} \geq 1 - \alpha \). Suppose that the distribution \( P_{\ast} \) is a mixture distribution with a density defined by (2). Then

\[
\Pr_{\sim P_{\ast}} \{ c(x, \xi) \leq 0 \} = \sum_{j=1}^{K} \pi_j \Pr_{\sim P_j} \{ c(x, \xi) \leq 0 \} .
\] (4)

To see this, we consider a simple property of GMM. Let \( H(\xi) \) be a function of \( \xi \). Suppose the expectation of \( H(\xi) \) under each \( P_j, j = 1, \ldots, K \) exists. Then

\[
E_{P_{\ast}} [H(\xi)] = \int H(z) \sum_{j=1}^{K} \pi_j p_j(z) dz = \sum_{j=1}^{K} \pi_j \int H(z)p_j(z) dz = \sum_{j=1}^{K} \pi_j E_{P_j} [H(\xi)],
\]

where the first and third equations follow from the definition of expectation, and the second equation holds since we can interchange the order of integral and finite sum. Then (4) follows from the above equation by setting \( H(\xi) = 1_{\{c(x,\xi)\leq0\}} \), which equals 1 if \( c(x, \xi) \leq 0 \) and 0 otherwise.

The formula (4) shows that the probability function with a mixture distribution can be expressed as the summation of probability functions with the underlying distributions being the component distributions. This result, from a crude simulation perspective, does not simplify the problem. However, it has interesting applications when each term in the summation can be simplified or handled efficiently. We will provide concrete examples to support the statement in the next section.

Problem (1a)-(1b) is a nonlinear optimization problem. Under the simulation based optimization framework, a natural approach of solving the problem is to use some nonlinear
optimization techniques. A nonlinear optimization algorithm typically requires the function values and the function gradients of the objective and constraints. For CCPs, these values are often difficult to be derived analytically and thus need to be estimated. In the literature, various methods have been proposed for estimating the gradient of the probability function, see, for instance, Prékopa (2003), Henrion (2012) and Henrion and Möller (2012). The difficulties of the estimation procedures of course depend on the structures of the probability function. But in general, estimating the gradient of a probability function is a challenging problem. In this paper, we analyze the gradient estimation for the chance constraint (1b). We now consider a GMM with density (3). We make the following assumption.

**Assumption 1** $\Pr \sim P_j \{c(x, \xi) \leq 0\}, j = 1, \cdots, K$ are continuously differentiable with respect to $x$ where $x \in X$.

Let $F_j(t, x)$ denote the cumulative distribution function of $c(x, \xi)$ under $P_j$. Suppose that $F_j(t, x)$ is continuously differentiable in $(t, x) \in (-\delta_j, \delta_j) \times X$ for some $\delta_j > 0$. Then Assumption 1 holds. Although Assumption 1 is a weak assumption, verifying it is not straightforward (see, e.g., Hong et al. (2011)). In the next section, we consider more specific settings and analyze the differentiability. Based on (4), we have the following proposition.

**Proposition 1** Suppose that the distribution $P_*$ is a GMM with density (3). Suppose that Assumption 1 is satisfied. Then $\Pr \sim P_*, \{c(x, \xi) \leq 0\}$ is differentiable with respect to $x$, and

$$
\nabla_x \Pr \sim P_* \{c(x, \xi) \leq 0\} = \sum_{j=1}^{K} \pi_j \nabla_x \Pr \sim P_j \{c(x, \xi) \leq 0\}.
$$

Proposition 1 shows that the gradient of the probability function with a mixture distribution is the weighted sum of the gradients of the probability functions with the component distributions. Suppose the gradient of the probability function with each component distribution can be handled efficiently. Then Proposition 1 guarantees that we can handle the gradient for the mixture model efficiently. In next section we will also give concrete examples that support the statement.

Proposition 1 also has applications in sensitivity analysis in risk management. In the sensitivity analysis of a probability function or a value-at-risk (VaR) risk measure, one may want to estimate the derivative of a probability or a VaR. If we use a mixture model to fit the financial data, we may want to implement Proposition 1 to simplify the problem. Of course, in some other probability optimization problems, e.g., optimizing a probability function, Proposition 1 may also be the support for algorithm development.

### 3 Solution Techniques for Different Scenarios

We have introduced CCPs with mixture distributions and have analyzed the model structures. In this section, we consider several scenarios in which the structured CCPs are used to model
various practical decisions. We mainly analyze how the chance constraint can be handled to embrace various optimization techniques.

### 3.1 Joint Chance Constrained Programs

We first consider a class of CCPs with the following chance constraint

$$\Pr_{\sim P^*} \{ B\xi \leq x \} \geq 1 - \alpha,$$

(5)

where $B$ is a $m \times k$ matrix and $x$ is the $m$-dimensional decision vector. The constraint (5) is a joint chance constraint given that $m > 1$. Let $b_i^T$ be the $i$-th row of $B$ and $x_i$ be the $i$-th component of $x$. By defining $c_i(x, \xi) = b_i^T \xi - x_i$, we see that (5) is an instance of (1b). In (5) the random vector $\xi$ and the decision vector $x$ are separated. This class of CCPs have been studied widely in the literature, see, e.g., van Ackooij et al. (2010), Henrion and Möller (2012), and van Ackooij et al. (2014). Henrion and Möller (2012) studied (5) and implemented the model in an electricity network capacity optimization problem. They considered the case where $P^*$ is a Gaussian distribution and studied how to estimate the gradient of the probability function in (5). As their main contribution, they showed that under the Gaussian distribution setting, calculations of the gradients can be reduced to calculations of the probability function values.

We now build a similar gradient formula for the probability function with the GMM. To ease the analysis, we define

$$F^j(x) = \Pr_{\sim P^j} \{ B\xi \leq x \}, \quad F^*(x) = \Pr_{\sim P^*} \{ B\xi \leq x \}.$$  

Theorem 3.3 (or Theorem 4.1) of Henrion and Möller (2012) provided conditions ensuring the differentiability of $F^j(x)$ and built an expression for $\nabla_x F^j(x)$. In Appendix A.1, we present their results. We thus take the gradient expression $\nabla_x F^j(x)$ as given. Following Henrion and Möller (2012), we introduce the notion of nondegeneracy for a linear inequality system. Consider the system $Bz \leq x$ with $B$ and $x$ given in (5). Let

$$I(B, x) = \left\{ I \subset \{1, \cdots, m\} \mid \exists z : b_i^T z = x_i \ (i \in I), b_i^T z < x_i \ (i \in \{1, \cdots, m\} \setminus I) \right\}.$$  

The system $Bz \leq x$ is called nondegenerate if $\text{rank} \{b_i\}_{i \in I} = \#I, \forall I \in I(B, x)$.

Based on Proposition 1, we can derive an expression for the gradient of $F^*(x)$. We summarize the result in the following proposition.

**Proposition 2** Suppose $P^*$ is a GMM defined by (3). Let $x$ be such that $Bz \leq x$ is nondegenerate. Then

$$F^*(x) = \sum_{j=1}^{K} \pi_j F^j(x); \quad \nabla_x F^*(x) = \sum_{j=1}^{K} \pi_j \nabla_x F^j(x),$$

where $\nabla_x F^j(x)$ is calculated according to Lemma 1 in Appendix A.1.

Proposition 2 generalizes the gradient formulas of Henrion and Möller (2012). Considering
that GMMs can model many distribution structures very well (e.g., McLachlan and Peel 2000, Wang and Taaffe 2015), such a generalization is very useful and can substantially extend the application of the approach proposed by Henrion and Möller (2012). Consider now the following CCP

\[
\begin{align*}
\min_{x \in X} & \quad h(x) \\
\text{subject to} & \quad \Pr_{\sim P_*} \{ B\xi \leq x \} \geq 1 - \alpha.
\end{align*}
\]

We can use some nonlinear optimization procedure as in Henrion and Möller (2012) to solve the problem. During the procedure, we can implement Proposition 2 to estimate the function value and the gradient for the chance constraint.

Proposition 2 provides a support for accessing the gradient of the probability function that possesses the linear separable structure. In the literature, there have also been some investigations for computing the gradient of the probability function for some nonlinear case with a Gaussian distribution, see, for instance, van Ackooij and Henrion (2014). For such nonlinear case, we can also implement Proposition 1 to assess the corresponding gradients and to conduct optimization accordingly.

The idea above can be adopted for another class of CCPs that have the separable structure. The corresponding chance constraint takes the following form.

\[
\Pr_{\sim P_*} \{ \xi \leq f(x) \} \geq 1 - \alpha,
\]

where \( f(x) = (f_1(x), \cdots, f_k(x))^\top \) with \( f_i(x), i = 1, \cdots, k \) being real valued functions defined on \( X \). Let \( \xi_{(i)} \) denote the \( i \)-th component of \( \xi \). By setting \( m = k \) and defining \( c_i(x, \xi) = \xi_{(i)} - f_i(x) \), the constraint above can be fit into (1b). The chance constraint was discussed in Prékopa (2003), Henrion (2012) and Henrion and Möller (2012). Under the Gaussian distribution \( \mathcal{N}(z | \mu, \Sigma) \) with \( \Sigma \) being positive definite, it has been built that calculations of \( \nabla_z \Pr_{\mathcal{N}(z | \mu, \Sigma)} \{ \xi \leq z \} \) can be reduced to calculations of the probability function values. Suppose that \( f(x) \) is continuously differentiable in \( x \) and \( P_* \) is a GMM defined by (3). Then \( \Pr_{\sim P_*} \{ \xi \leq f(x) \} \) is differentiable in \( x \). Similarly as in Proposition 2, we can convert the evaluations of the function values and the gradients for the probability function under the GMM to those under the Gaussian distribution. Then, the gradient-based nonlinear optimization approach can be implemented accordingly.

### 3.2 Single Chance Constrained Programs

We next consider a class of SCCPs with a linear product structure:

\[
\begin{align*}
\min_{x \in X} & \quad h(x) \\
\text{subject to} & \quad \Pr_{\sim P_*} \left\{ f_0(x) + f(x)^\top \xi \leq 0 \right\} \geq 1 - \alpha,
\end{align*}
\]
where \( f(x) = (f_1(x), \ldots, f_k(x))^\top \) and \( f_i(x), i = 0, 1, \ldots, k \) are real valued functions defined on \( X \). Constraint (6b) is also an instance of (1b) by setting \( m = 1 \) and \( c_1(x, \xi) = f_0(x) + f(x)^\top \xi \). In contrast to the separable structure considered in preceding subsection, the random vector \( \xi \) and the decision vector \( x \) in the chance constraint (6b) take a product form. This class of chance constraints has been studied widely in the literature, see, e.g., Nemirovski and Shapiro (2006), Chen et al. (2010), Zymler et al. (2013) and Hanasusanto et al. (2017). In our numerical experiments, we will study a typical mean-VaR portfolio selection model that can be written as an instance of (6a)-(6b). It is typically difficult to derive some deterministic representation for the chance constraint (6b). However, when the underlying distribution \( P^* \) falls in the normal distribution family, (6b) admits a simplified representation.

Specifically, suppose that the distribution \( P^* \) is \( \mathcal{N}(z | \mu, \Sigma) \). Then (6b) is equivalent to the following constraint

\[
\Phi^{-1}(1 - \alpha) \sqrt{f(x)^\top \Sigma f(x) + (f_0(x) + f(x)^\top \mu)^2} \leq 0,
\]

where \( \Phi^{-1}(\cdot) \) is the inverse of the standard normal distribution function, see, e.g., Henrion (2007). Since it is easy to evaluate \( \Phi^{-1}(\cdot) \), (7) can be treated as a deterministic constraint. Suppose further that \( h(x), f_0(x) \) and \( f(x) \) are linear in \( x \) and \( \alpha \leq 0.5 \). Then constraint (7) is known as a second-order cone constraint. In this setting, Problem (6a)-(6b) can be reformulated as a second-order conic program (SOCP) which can be solved by conventional optimization softwares, e.g., SeDuMi (Sturm (1999)).

In this subsection, we explore the structures of constraint (6b) under the GMM. Suppose that the distribution \( P^* \) takes the form of (3). Following from (4), (6b) is the same as

\[
\sum_{j=1}^{K} \pi_j \Pr_{P_j} \left\{ f_0(x) + f(x)^\top \xi \leq 0 \right\} \geq 1 - \alpha,
\]

where \( P_j = \mathcal{N}(z | \mu_j, \Sigma_j) \). Here we exclude the case where \( f(x) = 0 \), because when \( f(x) = 0 \), the constraint in the probability function does not involve any randomness. Then, the constraint above can be further written as

\[
\sum_{j=1}^{K} \pi_j \Phi \left( \frac{-[f_0(x) + f(x)^\top \mu_j]}{\sqrt{f(x)^\top \Sigma_j f(x)}} \right) \geq 1 - \alpha
\]

where \( \Phi(\cdot) \) is the standard normal distribution function.

3.2.1 A Gradient-based Algorithm

Because we reformulate (6b) as a deterministic constraint (9), it is natural to implement a gradient-based optimization algorithm. We assume that \( f_0(x) \) and \( f(x) \) are continuously differentiable in \( x \). Let

\[
u_j(x) = f_0(x) + f(x)^\top \mu_j, \quad \tilde{u}_j(x) = \sqrt{f(x)^\top \Sigma_j f(x)}.
\]
Note that the gradient (derivative) of $\Phi(\cdot)$ is simply the density function, denoted as $\phi(\cdot)$, of the standard normal distribution. Then the left hand side of (9) is differentiable in $x$ and the gradient is

$$
\sum_{j=1}^{K} \pi_j \phi \left( \frac{-u_j(x)}{\tilde{u}_j(x)} \right) \frac{-\nabla u_j(x) \tilde{u}_j(x) + u_j(x) \nabla \tilde{u}_j(x)}{\tilde{u}_j^2(x)}.
$$

Therefore, the values and gradients of the function in (9) can be evaluated efficiently. It then allows us to use conventional gradient-based nonlinear optimization algorithms for the problem.

A recent study by Peña-Ordieres et al. (2019) investigated a quantile reformulation of the CCP. Peña-Ordieres et al. (2019) transformed the chance constraint into a quantile (or a VaR) constraint. They then considered a sample approximation and a smooth approximation to construct some smooth constraint to approximate the quantile constraint, and then implemented some nonlinear optimization procedure to solve the approximation problem. Their findings are that in terms of numerical computation, the quantile reformulation shows certain superiority. Recall that a $1 - \alpha$ VaR of a random loss $\eta$ with distribution $P_\ast$ is defined as

$$
\text{VaR}_{1 - \alpha, P_\ast}(\eta) := \inf \{ v \in \mathbb{R} : \Pr_{\sim P_\ast} \{ \eta \leq v \} \geq 1 - \alpha \}.
$$

Following the idea of Peña-Ordieres et al. (2019), we can solve the following VaR constrained equivalent formulation for Problem (6a)-(6b).

\begin{align*}
\text{minimize} & \quad h(x) \quad \text{(10a)} \\
\text{subject to} & \quad \text{VaR}_{1 - \alpha, P_\ast}(f_0(x) + f(x)^\top \xi) \leq 0. \quad \text{(10b)}
\end{align*}

Let $q(x) = \text{VaR}_{1 - \alpha, P_\ast}(f_0(x) + f(x)^\top \xi)$. We show that under our GMM and linear product setting, we can directly derive the gradient of $q(x)$. Following from the definition of quantile (or VaR), we have

$$
\sum_{j=1}^{K} \pi_j \Phi \left( \frac{q(x) - [f_0(x) + f(x)^\top \mu_j]}{\sqrt{f(x)^\top \Sigma_j f(x)}} \right) = 1 - \alpha. \quad (11)
$$

Given $x$, since $\Phi(\cdot)$ is strictly increasing, we can evaluate $q(x)$ using a bisection search method. To derive the gradient of $q(x)$, we differentiate both sides of (11) with respect to $x$ and apply the implicit function theorem. This yields

$$
\nabla q(x) = \left[ \sum_{j=1}^{K} \pi_j \phi \left( \frac{q(x) - u_j(x)}{\tilde{u}_j(x)} \right) \right]^{-1} g(x)
$$

where

$$
g(x) = \sum_{j=1}^{K} \pi_j \phi \left( \frac{q(x) - u_j(x)}{\tilde{u}_j(x)} \right) \left( \nabla u_j(x) \tilde{u}_j(x) + (q(x) - u_j(x)) \nabla \tilde{u}_j(x) \right) \frac{u_j^2(x)}{\tilde{u}_j^2(x)}.
$$

Since we can evaluate $q(x)$ and its gradient $\nabla q(x)$, we can now implement a nonlinear optimization procedure to solve Problem (10a)-(10b). Note that Peña-Ordieres et al. (2019)
considered a very general setting and thus they need to use sample approximation and smooth approximation. In our setting, we first conduct an input modeling using GMM. Then for the more specific GMM and linear product setting, our problem become much easier to handle.

3.2.2 A Spatial Branch-and-Bound Algorithm

Due to the non-convexity of CCPs, nonlinear optimization procedures typically could only guarantee stationary points for the CCPs. However, the structure of (9) offers some potential for searching the global optimal solutions for the CCPs. In the rest of this section, we explore the structure of the CCP under the GMM and develop a global optimization procedure. We first build the following result.

**Proposition 3** Suppose that the distribution $P_\ast$ is a GMM defined by (3). Then Problem (6a)-(6b) is equivalent to the following problem

\[
\begin{align*}
\text{minimize} & \quad h(x) \\
\text{subject to} & \quad \Phi^{-1}(y_j)\sqrt{f(x)^\top \Sigma_j f(x)} + \left(f_0(x) + f(x)^\top \mu_j\right) \leq 0, \ j = 1, \cdots, K, \\
& \quad \sum_{j=1}^K \pi_j y_j \geq 1 - \alpha, \ 0 \leq y_j \leq 1, \ j = 1, \cdots, K.
\end{align*}
\]

**Proof.** By introducing an auxiliary decision vector $y = (y_1, \cdots, y_K)^\top$, we can transform constraint (8) equivalently to the following set of constraints

\[
\Pr_{\sim P_j} \left\{ f_0(x) + f(x)^\top \xi \leq 0 \right\} \geq y_j, \ j = 1, \cdots, K,
\]

\[
\sum_{j=1}^K \pi_j y_j \geq 1 - \alpha, \ 0 \leq y_j \leq 1, \ j = 1, \cdots, K.
\]

Since $P_j$ is $\mathcal{N}(z| \mu_j, \Sigma_j)$, we have (13) are equivalent to (12b). This concludes the proof of the proposition.

The equivalence here guarantees that if $(x^\ast, y^\ast)$ is an optimal solution of Problem (12a)-(12c), then $x^\ast$ is optimal to Problem (6a)-(6b). Problem (12a)-(12c) becomes a deterministic optimization problem with some second-order cone type structure. A major difficulty for handling the problem is that both $x$ and $y$ are decision variables. To utilize the second-order cone type structure, one potential approach is to branch the decision variables $y$.

Note that for the setting, one tractable case is when $K = 1$, i.e., the underlying distribution $P_\ast$ degenerates to a Gaussian distribution. With this setting, as discussed in the preceding subsection, we can transform (12a)-(12c) into a deterministic optimization problem which has constraint (7). In particular, when $h(x)$, $f_0(x)$ and $f(x)$ are all linear in $x$ and $\alpha \leq 0.5$, the resulting problem is an SOCP.

In this subsection, we assume that $f_0(x)$ and $f(x)$ are linear in $x$ but allow the objective function $h(x)$ to be linear or nonlinear. According to the special structure of the mixture model,
we propose a BB type procedure to handle Problem (12a)-(12c). We first analyze the bound of the decision vector $y$ for Problem (12a)-(12c) being feasible. Note that for $j = 1, \ldots, K$,

$$1 - \alpha \leq \pi_j y_j + \sum_{i \neq j}^K \pi_i y_i \leq \pi_j y_j + \sum_{i \neq j}^K \pi_i = \pi_j y_j + (1 - \pi_j).$$

It follows that $y_j \geq 1 - \alpha / \pi_j$. Let $y_j^0 = \max\{0, 1 - \alpha / \pi_j\}$ and $\overline{y}_j^0 = 1$. Thus we only need to consider the following region

$$\Delta^0 := \{y \in \mathbb{R}^K \mid y_j^0 \leq y_j \leq \overline{y}_j^0, j = 1, \ldots, K\}.$$

The basic idea of our BB algorithm is to partition the region $\Delta^0$ into sub-rectangles and consider a relaxation of the original problem over a given sub-rectangle. Specifically, consider a sub-rectangle of $\Delta^0$:

$$\Delta := \{y \in \mathbb{R}^K \mid y_j \leq y_j \leq \overline{y}_j, j = 1, \ldots, K\}.$$

Restricting Problem (12a)-(12c) to $\Delta$, we obtain the following problem.

**minimize**

$$h(x)$$

**subject to**

$$\Phi^{-1}(y_j)\sqrt{f(x)^\top \Sigma_j f(x)} + (f_0(x) + f(x)^\top \mu_j) \leq 0, j = 1, \ldots, K,$$

$$\sum_{j=1}^K \pi_j y_j \geq 1 - \alpha.$$

Notice that $\Phi^{-1}(\cdot)$ is an increasing function and $\sqrt{f(x)^\top \Sigma_j f(x)}$ is non-negative. We consider the following relaxation of Problem (14a)-(14c):

**minimize**

$$h(x)$$

**subject to**

$$\Phi^{-1}(y_j)\sqrt{f(x)^\top \Sigma_j f(x)} + (f_0(x) + f(x)^\top \mu_j) \leq 0, j = 1, \ldots, K,$$

$$\sum_{j=1}^K \pi_j y_j \geq 1 - \alpha.$$

At each iteration of our BB method, the domain of the vector $y$ is partitioned into sub-rectangles. Solving (15a)-(15c) provides a lower bound for the optimal value of Problem (14a)-(14c). If the optimal solution is also feasible to (12a)-(12c), then its objective value provides an upper bound for the optimal value of (12a)-(12c). At each iteration, a new problem (15a)-(15c) on a certain sub-rectangle $\Delta$ is solved and a better solution could possibly be identified. For any problem (15a)-(15c) that is infeasible or its optimal value is greater than or equal to the best attained objective value of (12a)-(12c), the corresponding sub-rectangle is removed from further consideration. We then select one of the remaining active sub-rectangles, under
some selection criterion, and partition it into two sub-rectangles for further consideration. This
process repeats until all the sub-rectangles are removed. The method is now formally described
as follows.

Algorithm 1

Step 0 (Initialization): Set \( i := 0 \), \( \Delta^i := \{ y \in \mathbb{R}^K \mid y_j^0 \leq y_j \leq y_j^0, j = 1, \ldots, K \} \), \( v^* := \infty \), \( \underline{v} := -\infty \), \( flag := 0 \). Solve Problem (15a)-(15c) with \( \Delta := \Delta^i \).

IF it is infeasible, Problem (12a)-(12c) is infeasible, STOP;
ELSE, denote its optimal solution and optimal objective value as \((x^c, y^c)\) and \(v^c\), respectively.

IF \( x^c \) is not a feasible solution to Problem (12a)-(12c), \((x^i, y^i) := (x^c, y^c)\), \( v^i := v^c \), \( \underline{v} := \underline{v} \), \( \Omega := \{\Delta^i, v^i\} \). Set \( i := i + 1 \), GOTO Step 1;
ELSE, \( v^* := v^c \), \( x^* := x^c \), \( fflag := 1 \), \( x^* \) is an optimal solution to Problem (12a)-(12c), STOP.

END IF

Step 1:

IF \( \Omega \neq \emptyset \), GOTO Step 2;
ELSE IF \( fflag = 0 \), Problem (12a)-(12c) is infeasible, STOP;
ELSE, \( x^* \) is an optimal solution to Problem (12a)-(12c), STOP.
END IF

Step 2: Choose and remove instance \{\(\Delta^i, v^i\)\} from \(\Omega\) with the smallest \(v^i\), denoted as \(v^i_s\). If more than one rectangles have the smallest \(v^i\), randomly choose one.

Subdivide \(\Delta^i\) into two rectangles \(\Delta^i_1\) and \(\Delta^i_2\) by dividing an edge of \(\Delta^i\) at its midpoint according to some partition rule. Denote by \(\Delta^i_1\) the “upper” rectangle that includes the upper corner \((\overline{y}_1, \ldots, \overline{y}_K)\) of \(\Delta^i\) and by \(\Delta^i_2\) the “lower” rectangle that includes the lower corner \((\underline{y}_1, \ldots, \underline{y}_K)\) of \(\Delta^i\).

IF \( \sum_{j=1}^K \pi_j \overline{y}_j^1 > 1 - \alpha \) where \((\overline{y}_1^1, \ldots, \overline{y}_K^1)\) is the lower corner of \(\Delta^i_1\), set \(v^i_2 := v^i, \Omega := \Omega \cup \{\Delta^i_2, v^i_2\} \). Set \( i := i + 1 \), GOTO Step 1;
ELSE IF \( \sum_{j=1}^K \pi_j \overline{y}_j^2 < 1 - \alpha \) where \((\overline{y}_1^2, \ldots, \overline{y}_K^2)\) is the upper corner of \(\Delta^i_2\), solve Problem (15a)-(15c) with \(\Delta := \Delta^i_1\).

IF it is infeasible, \(\Omega := \Omega\). Set \( i := i + 1 \), GOTO Step 1;
ELSE, denote its optimal solution and optimal objective value as \((x^c, y^c)\) and \(v^c\), respectively.

IF \( v^c \geq v^* \), \(\Omega := \Omega\). Set \( i := i + 1 \), GOTO Step 1;
ELSE IF $v^c < v^*$ and $x^c$ is not a feasible solution to Problem (12a)-(12c), $(x^i, y^i) := (x^c, y^c)$, $v^i = v^c$, $g^i_1 := v^i$, $\Omega := \Omega \cup \{\Delta^i_1, g^i_1\}$. Set $i := i + 1$, GOTO Step 1;

ELSE, $v^* := v^c$, $x^* := x^c$, $flag := 1$, delete from $\Omega$ all instances $\{\Delta^i, g^i\}$ with $g^i \geq v^*$. Set $i := i + 1$, GOTO Step 1.

END IF

END IF

ELSE, set $g^i_2 := g^i_s$. Solve Problem (15a)-(15c) with $\Delta := \Delta^i_1$.

IF it is infeasible, $\Omega := \Omega \cup \{\Delta^i_2, g^i_2\}$. Set $i := i + 1$, GOTO Step 1;

ELSE, denote its optimal solution and optimal objective value as $(x^c, y^c)$ and $v^c$, respectively.

IF $v^c \geq v^*$, $\Omega := \Omega \cup \{\Delta^i_2, g^i_2\}$. Set $i := i + 1$, GOTO Step 1;

ELSE IF $v^c < v^*$ and $x^c$ is not a feasible solution to Problem (12a)-(12c), $(x^i, y^i) := (x^c, y^c)$, $v^i = v^c$, $g^i_1 := v^i$, $\Omega := \Omega \cup \{\Delta^i_1, g^i_1\} \cup \{\Delta^i_2, g^i_2\}$. Set $i := i + 1$, GOTO Step 1;

ELSE, $v^* := v^c$, $x^* := x^c$, $flag := 1$, delete from $\Omega$ all instances $\{\Delta^i, g^i\}$ with $g^i \geq v^*$. Set $i := i + 1$, GOTO Step 1.

END IF

END IF

END IF

We provide more details for Algorithm 1. In Step 2, we only solve Problem (15a)-(15c) over upper rectangle $\Delta^i_1$. Problem (15a)-(15c) over lower rectangle $\Delta^i_2$ and the corresponding problem over $\Delta^i$ (note that $\Delta^i$ is divided to $\Delta^i_1$ and $\Delta^i_2$) share the same optimal value (if it exists). The reason is that the lower rectangle $\Delta^i_2$ shares the same lower corner with $\Delta^i$ so that the constraints (15b) are the same in both problems. Thus we can set $g^i_s$ as the lower bound of $\Delta^i_2$, without solving for it.

In Step 2, for the partition rule, we first choose the longest edge among edges whose lower bounds are less than 0.5 and midpoints are larger than 0.5. If we cannot find such an edge, we divide the longest edge among all edges of $\Delta^i$. Furthermore, if more than one edges can be chosen, we break the tie by randomly choosing one of them. Note that in Problem (12a)-(12c), we can obtain an optimal solution by restricting $y$ to the hyperplane $\sum_{j=1}^{K} \pi_j y_j = 1 - \alpha$. Thus, as in Step 2, we only consider rectangles that intersect with this hyperplane. This can be seen clearly in Figure 1. For the three rectangles in the figure, we only consider the rectangle $\Delta^2$ that intersects with the line (hyperplane) $\sum_{j=1}^{K} \pi_j y_j = 1 - \alpha$.

Algorithm 1 above can be regarded as a specific implementation of a general BB method within the framework described by Horst (1986) and Horst et al. (1995). Notice that at Step 2, we have $int\{\Delta^i_1\} \cap int\{\Delta^i_2\} = \emptyset$ and $\Delta^i = \Delta^i_1 \cup \Delta^i_2$, where $int(S)$ denotes the interior of set $S$. Algorithm 1 will generate lower bounds for the optimal value of Problem (12a)-(12c). When
it generates a feasible solution of Problem (12a)-(12c), the function value of the solution is an upper bound for the optimal value of Problem (12a)-(12c). From the algorithm we can see that the lower bound is tracked by the sequence \( \{ v_s^i \} \) and the upper bound is tracked by \( v^* \). It is easy to see that if Algorithm 1 terminates after a finite number of iterations at Step 1, then it achieves an optimal solution to Problem (12a)-(12c) or verifies the infeasibility of Problem (12a)-(12c). The following proposition gives the global convergent property of Algorithm 1.

**Proposition 4** Suppose that \( X \) is compact, and the functions \( h(x) \), \( f_0(x) \) and the vector-valued function \( f(x) \) are continuous and bounded on \( X \). Suppose that \( x_s^i \) produces \( v_s^i \), i.e., \( v_s^i = h(x_s^i) \). Denote the optimal value of Problem (12a)-(12c) by \( v_{\text{opt}} \). If Algorithm 1 generates an infinite sequence \( \{ v_s^i \} \), then \( v_s^i \rightarrow v_{\text{opt}} \), and any accumulation point \( x^* \) of \( \{ x_s^i \} \) is an optimal solution to Problem (12a)-(12c).

**Proof.** Because the algorithm constantly improves the lower bound by partition, the sequence \( \{ v_s^i \} \) is nondecreasing. Since \( v_s^i \) is always bounded above by \( v_{\text{opt}} \), \( v_s^i \) converge to some value \( v_s \leq v_{\text{opt}} \). We next show the opposite inequality. Consider the rectangle \( \Delta_i \) that generates \( v_s^i \). For simplicity we denote the lower corner of \( \Delta_i \) as \( y_s^i \). Suppose \( \{ x_{s,i}^i \}_{i \in \mathcal{N}} \) is a subsequence of \( \{ x_s^i \}_{i \in \mathcal{N}} \) converging to \( x^* \). Note that \( \{ y_s^i \} \) is a bounded sequence. It has convergent subsequence. Thus by passing to a subsequence if necessary, we can assume that \( \{ (x_{s,i}^i, y_s^i) \}_{i \in \mathcal{N}} \) converge to \( (x^*, y^*) \) for some \( y^* \). By the continuity of \( h(x) \), we obtain \( h(x^*) = v_s \leq v_{\text{opt}} \). We show that \( (x^*, y^*) \) is a feasible solution of Problem (12a)-(12c). First, by the partition rule in Step 2, \( y^* \) will be on the hyperplane \( \sum_{j=1}^K \pi_j y_j = 1 - \alpha \).

Denote

\[
F_j(x, y_j) = \Phi^{-1}(y_j) \sqrt{f(x)^\top \Sigma_j f(x) + (f_0(x) + f(x)^\top \mu_j)}.
\]

We have \( F_j(x_{s,i}^i, y_{s,i}^i) \leq 0 \). Since \( f_0(x) \), \( f(x) \) and \( \Phi^{-1}(y_j) \) are continuous, \( F_j(x, y_j) \) is continuous. Letting \( i_h \rightarrow \infty \), we have \( F_j(x^*, y^*_j) \leq 0 \). This indicates \( (x^*, y^*) \) is a feasible solution of Problem (12a)-(12c). Thus \( h(x^*) \geq v_{\text{opt}} \). This concludes the proof of the proposition. \[\square\]
Now let us further analyze Algorithm 1. In each iteration of the algorithm, we need to solve a subproblem. The subproblems are still nonlinear optimization problems. Therefore, we need to call some softwares or design appropriate procedures to solve the problems. Note that the functions $f_0$ and $f$ are all linear in $x$. Then Problem (15a)-(15c) has a conic program structure. We consider a number of cases, depending on whether the problem is convex. For the convexity we have the following proposition.

**Proposition 5** Suppose that $f_0(x)$ and $f(x)$ are linear in $x$. If $y_j \geq 1/2$ for $j = 1, \ldots, K$, then (15b) is a second-order cone constraint. Moreover, if $y_j^0 \geq 1/2$, $j = 1, \ldots, K$, or equivalently, $\pi_j \geq 2\alpha$, $j = 1, \ldots, K$, then (15b) is always a second-order cone constraint.

Suppose that the conditions in Proposition 5 are satisfied. If $h(x)$ is linear in $x$, Problem (15a)-(15c) is an SOCP. Note that the linear case has been studied widely in the literature and management practice. If $h(x)$ is convex quadratic in $x$, Problem (15a)-(15c) is a convex problem with quadratic objective and second-order cone constraint. For the two settings we can use conventional softwares, e.g., SeDuMi (Sturm (1999)), to solve Problem (15a)-(15c) efficiently. If $h(x)$ is a more general convex function, Problem (15a)-(15c) is a more general convex problem and a typical software can solve it to global optimality easily. In these cases, Algorithm 1 only needs to solve a sequence of convex optimization problems. Our numerical experiments indicate that Algorithm 1 is typically fast for the convex case. Note that in real applications, $\alpha$ usually takes a small value, e.g., 0.05. Then, it is a common case where $\pi_j \geq 2\alpha$, $j = 1, \ldots, K$, when $K$ is small. In the more general case where convexity cannot be preserved, we can transform Problem (15a)-(15c) to an optimization problem with quadratic constraints. Then, we can use some commercial softwares, e.g., BARON (see, e.g., Tawarmalani and Sahinidis (2005) and Sahinidis (2017)), to solve the problem globally. It is worthwhile noting that for Problem (12a)-(12c), the number of branch variables, which is the number of mixture components $K$, is usually very small. Thus the BB method can often work efficiently.

Finally, we remark that our approach may also handle some CCPs that have the same structure as (6a)-(6b) but part of the decision variables are integer-valued, that is, $x \in Z^{d_1} \times \mathbb{R}^{d_2}$, where $Z$ is the set of integers and $d_1 + d_2 = d$. The model imposes that the first $d_1$ elements of $x$ are integer-valued and the rest $d_2$ elements are continuous. In the applications, if the integer variables are 0-1 valued, then one can instead impose $x \in \{0, 1\}^{d_1} \times \mathbb{R}^{d_2}$. Such structured models are called mixed integer CCPs. Mixed integer CCPs have received much attention and have been used to model various real decisions, see, e.g., Ahmed and Papageorgiou (2013) and Xie and Ahmed (2018). Bringing in integer variables can extend the application of the linear product form CCP. For instance, it can model a mean-VaR portfolio selection problem with cardinality constraint (e.g., Gao and Li (2013)) and the problem with probabilistic knapsack constraints (e.g., Atamtürk et al. (2013)). When integer variables enter Problem (6a)-(6b), our approach succeeds to convert a stochastic mixed integer program to a deterministic mixed integer program. For the linear or quadratic objective function, Problem (15a)-(15c) is a mixed integer quadratic program. We can implement the integer optimization software that is capable of dealing with the integer variables. In this setting, we need to rely on the softwares to explore
and deal with the integer structures, and the efficiency will depend on the power of softwares on dealing with integer variables.

4 Numerical Illustrations

In this section, we first test our algorithm on a CCP with a linear product form, where the parameters are randomly generated. We then implement our approach to solve a portfolio selection problem with risk control on VaR, which is a widely used risk measure in the financial industry, see, for instance, Jorion (2006). Throughout the experiments, the algorithms are coded in Matlab R2019a and run on an Intel Core i5 PC with 3.31 GHz and 8 GB RAM. For the BB algorithm, we use SeDuMi 1.05 to solve the subproblems if they are SOCPs. If the subproblems are not SOCPs, we call the software BARON 17.10.16. (see Tawarmalani and Sahinidis (2005) and Sahinidis (2017)) to solve them.

4.1 A Test Problem

We consider a specific instance of Problem (6a)-(6b). As discussed in Section 3.2, Problem (6a)-(6b) can be transformed as

\[
\begin{align*}
\text{minimize} & \quad h(x) \\
\text{subject to} & \quad \sum_{j=1}^{K} \pi_j \Phi \left( \frac{-[f_0(x) + f(x)^\top \mu_j]}{\sqrt{f(x)^\top \Sigma_j f(x)}} \right) \geq 1 - \alpha.
\end{align*}
\]  

In this specific instance, we set \(h(x)\) to be a linear function \(c^\top x\), \(X\) to be \([0, 5]^d\), \(f_0(x)\) to be a constant function, and \(f(x) = (f_1(x), \ldots, f_K(x))^\top\) to be a linear function vector \(a + Bx\) where \(a = (0, \ldots, 0)^\top\) and \(B\) is an identity matrix. In the experiment, we randomly generate the coefficients of the objective function \(h(x)\) and the parameters of the GMM as follows: (i) the elements of \(c\) are generated from the \([-10, 0]\) uniform distribution independently; (ii) the elements of \(\mu_j\) (\(j = 1, \ldots, K\)) are respectively generated from the \([0, 5]\) uniform distribution independently; (iii) \(\Sigma_j\) (\(j = 1, \ldots, K\)) are generated independently from \(RR^\top\) where \(R\) is a \(k\)-dimensional column vector and the elements of \(R\) independently follow the \([0, 1]\) uniform distribution; (iv) \(\pi_j = r_j / \left( \sum_{i=1}^{K} r_i \right) \) (\(j = 1, \ldots, K\)) where \(r_j\) is generated independently from the \([0, 1]\) uniform distribution. We consider two scenarios where \(\alpha = 0.05\) and \(\alpha = 0.1\). For \(\alpha = 0.05\), we set \(f_0(x) = -2000\) and ensure that \(\pi_j \geq 2\alpha, j = 1, \ldots, K\) such that all the subproblems are convex, whereas for \(\alpha = 0.1\) we set \(f_0(x) = -500\) and make sure that \(\pi_j \geq 2\alpha\) is violated for at least one \(j\) such that some of the subproblems are nonconvex. For each combination of \(d\) and \(K\) we randomly generate the input parameters above five times. For the simulated data, we use both the nonlinear optimization algorithm (Matlab fmincon function) and the BB algorithm (Algorithm 1) to solve Problem (16a)-(16b) respectively. The aim is to learn the features and properties of both algorithms.

For the nonlinear optimization approach, we choose to solve the quantile reformulation. As
in Section 3.2, we embed the gradient $\nabla q(x)$ in fmincon to solve the problem. For fmincon, we set the maximal number of iterations to be $10^4$ and the tolerance on the constraint violation to be $10^{-4}$. The BB algorithm is terminated if $|U - L|/|L| \leq 10^{-4}$ where $U$ and $L$ are respectively the incumbent upper bound and lower bound for Problem (16a)-(16b). In order to speed up the BB algorithm, we prune the subproblems for which $||\bar{y} - y|| < 10^{-2}$ where $|| \cdot ||$ denotes the Euclidean norm. For BARON, we set the relative termination tolerance to be $10^{-3}$ which can ensure the accuracy of our solutions when objective function values range from $[100, 2000]$.

We also set the parameters “DeltaTerm” and “DeltaA” in BARON to 1 so that BARON will terminate if the value of the optimal value is not improved by at least 1 for a period of time set by default in BARON. The remaining parameters are set by default in BARON. We initialize BARON and the fmincon function with the same starting point $(0.5, \cdots, 0.5)^T$.

The optimal values under $\alpha = 0.05$ and $\alpha = 0.1$ derived by the two procedures are reported in Tables 1 and 3, where the three columns “Ave”, “Min” and “Max” represent the average, minimal and maximal optimal values among the five runs obtained by the fmincon function and the BB algorithm under each combination of $K$ and $d$. From Tables 1 and 3, we can see that the values returned by the fmincon function are not globally optimal. The BB algorithm can significantly improve the values to the global optimality. Tables 2 and 4 report the computational times for both algorithms to obtain the corresponding optimal values. The tables show that the computational time of the fmincon function is relatively stable, whereas the computational time of the BB algorithm increases with $K$ and $d$. For the case where $\alpha = 0.05$, $\alpha = 0.1$.

| $K$ | $d$ | BB Algorithm | | | fmincon | | | | | Ave | Min | Max | Ave | Min | Max |
|-----|-----|--------------|-----|-----|-------|-----|-----|-----|-----|-----|-----|-----|-----|-----|
| 2   | 50  | -1292.313    | -1439.261 | -1122.957 | -266.525 | -280.969 | -253.745 | | | | | | |
|     | 80  | -1511.231    | -1712.534 | -1376.577 | -412.855 | -452.296 | -388.139 | | | | | | |
|     | 100 | -1490.221    | -1550.129 | -1434.198 | -489.141 | -509.706 | -469.895 | | | | | | |
|     | 200 | -1312.489    | -1352.622 | -1294.549 | -993.180 | -1016.516 | -977.100 | | | | | | |
|     | 100 | -1527.192    | -1610.444 | -1435.518 | -487.827 | -515.314 | -467.040 | | | | | | |
|     | 200 | -1305.101    | -1319.046 | -1276.179 | -964.811 | -983.476 | -939.194 | | | | | | |
| 4   | 50  | -1218.152    | -1360.937 | -1075.455 | -255.575 | -272.517 | -246.051 | | | | | | |
|     | 80  | -1519.981    | -1643.553 | -1425.657 | -393.424 | -421.265 | -374.130 | | | | | | |
|     | 100 | -1499.559    | -1528.357 | -1463.694 | -485.954 | -497.402 | -481.082 | | | | | | |
|     | 200 | -1301.342    | -1329.682 | -1292.973 | -955.121 | -969.243 | -920.881 | | | | | | |
| 5   | 50  | -1189.390    | -1269.474 | -1076.447 | -250.423 | -266.229 | -234.681 | | | | | | |
|     | 80  | -1547.467    | -1523.690 | -1391.711 | -384.544 | -390.050 | -381.163 | | | | | | |
|     | 100 | -1512.267    | -1551.571 | -1483.883 | -484.757 | -490.415 | -480.607 | | | | | | |
|     | 200 | -1305.836    | -1335.323 | -1267.842 | -951.073 | -969.243 | -920.881 | | | | | | |

Table 1: Optimal values obtained by the nonlinear optimization algorithm (fmincon) and the BB algorithm for Problem (16a)-(16b) with $\alpha = 0.05$.
Table 2: Computational times of the nonlinear optimization algorithm (fmincon) and the BB algorithm for Problem (16a)-(16b) with $\alpha = 0.05$

<table>
<thead>
<tr>
<th>$K$</th>
<th>$d$</th>
<th>BB Algorithm</th>
<th>fmincon</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td></td>
<td>Ave (s) Min (s) Max (s)</td>
<td>Ave (s) Min (s) Max (s)</td>
</tr>
<tr>
<td>2</td>
<td>50</td>
<td>0.485 0.376 0.846</td>
<td>3.963 3.782 4.235</td>
</tr>
<tr>
<td></td>
<td>80</td>
<td>0.534 0.449 0.730</td>
<td>3.000 2.848 3.118</td>
</tr>
<tr>
<td></td>
<td>100</td>
<td>0.765 0.538 0.971</td>
<td>2.990 2.687 3.243</td>
</tr>
<tr>
<td></td>
<td>200</td>
<td>2.054 1.702 2.549</td>
<td>2.152 2.031 2.294</td>
</tr>
<tr>
<td>3</td>
<td>50</td>
<td>2.561 1.855 3.949</td>
<td>4.252 4.012 4.543</td>
</tr>
<tr>
<td></td>
<td>100</td>
<td>5.311 4.030 6.442</td>
<td>3.294 3.128 3.645</td>
</tr>
<tr>
<td></td>
<td>200</td>
<td>18.773 12.816 23.027</td>
<td>2.622 2.588 2.659</td>
</tr>
<tr>
<td>4</td>
<td>50</td>
<td>7.655 5.546 11.165</td>
<td>5.339 4.840 5.849</td>
</tr>
<tr>
<td></td>
<td>80</td>
<td>22.951 20.400 27.588</td>
<td>3.966 3.556 4.260</td>
</tr>
<tr>
<td></td>
<td>100</td>
<td>27.338 14.268 36.606</td>
<td>3.733 3.577 3.848</td>
</tr>
<tr>
<td></td>
<td>200</td>
<td>109.866 77.183 152.229</td>
<td>2.986 2.897 3.084</td>
</tr>
<tr>
<td>5</td>
<td>50</td>
<td>23.994 12.730 34.068</td>
<td>5.864 5.214 6.175</td>
</tr>
<tr>
<td></td>
<td>80</td>
<td>90.261 29.365 124.705</td>
<td>4.507 4.329 4.774</td>
</tr>
<tr>
<td></td>
<td>100</td>
<td>127.889 92.362 179.351</td>
<td>4.518 4.094 4.862</td>
</tr>
<tr>
<td></td>
<td>200</td>
<td>546.763 361.746 694.797</td>
<td>3.475 3.325 3.636</td>
</tr>
</tbody>
</table>

Table 3: Optimal values obtained by the nonlinear optimization algorithm (fmincon) and the BB algorithm for Problem (16a)-(16b) with $\alpha = 0.1$

<table>
<thead>
<tr>
<th>$K$</th>
<th>$d$</th>
<th>BB Algorithm</th>
<th>fmincon</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td></td>
<td>Ave Min Max</td>
<td>Ave Min Max</td>
</tr>
<tr>
<td>2</td>
<td>20</td>
<td>-562.368 -515.724 -485.280</td>
<td>-183.016 -207.276 -144.594</td>
</tr>
<tr>
<td></td>
<td>40</td>
<td>-626.106 -665.585 -599.190</td>
<td>-262.620 -273.820 -245.549</td>
</tr>
<tr>
<td></td>
<td>50</td>
<td>-633.679 -680.523 -604.856</td>
<td>-296.914 -317.853 -278.201</td>
</tr>
<tr>
<td>3</td>
<td>20</td>
<td>-494.604 -595.203 -424.636</td>
<td>-165.676 -185.190 -143.733</td>
</tr>
<tr>
<td></td>
<td>40</td>
<td>-625.063 -703.532 -567.163</td>
<td>-238.498 -260.696 -210.017</td>
</tr>
<tr>
<td></td>
<td>50</td>
<td>-633.679 -680.523 -604.856</td>
<td>-296.914 -317.853 -278.201</td>
</tr>
<tr>
<td></td>
<td>30</td>
<td>-570.632 -656.081 -529.852</td>
<td>-189.754 -213.799 -176.238</td>
</tr>
<tr>
<td></td>
<td>40</td>
<td>-613.218 -656.723 -545.359</td>
<td>-227.440 -245.802 -208.123</td>
</tr>
<tr>
<td>5</td>
<td>20</td>
<td>-504.371 -587.385 -441.662</td>
<td>-169.846 -175.948 -166.552</td>
</tr>
<tr>
<td></td>
<td>40</td>
<td>-601.461 -675.396 -502.529</td>
<td>-233.970 -248.218 -212.573</td>
</tr>
<tr>
<td></td>
<td>50</td>
<td>-574.048 -647.547 -537.995</td>
<td>-267.855 -303.001 -241.142</td>
</tr>
</tbody>
</table>
Table 4: Computational time of the nonlinear optimization algorithm (fmincon) and the BB algorithm for Problem (16a)-(16b) with $\alpha = 0.1$

<table>
<thead>
<tr>
<th>$K$</th>
<th>$d$</th>
<th>BB Algorithm Ave (s)</th>
<th>BB Algorithm Min (s)</th>
<th>BB Algorithm Max (s)</th>
<th>fmincon Ave (s)</th>
<th>fmincon Min (s)</th>
<th>fmincon Max (s)</th>
</tr>
</thead>
<tbody>
<tr>
<td>2</td>
<td>20</td>
<td>8.254</td>
<td>5.874</td>
<td>11.442</td>
<td>8.746</td>
<td>7.808</td>
<td>10.481</td>
</tr>
<tr>
<td></td>
<td>30</td>
<td>33.101</td>
<td>2.490</td>
<td>59.841</td>
<td>7.678</td>
<td>7.139</td>
<td>8.089</td>
</tr>
<tr>
<td></td>
<td>40</td>
<td>137.464</td>
<td>4.791</td>
<td>238.588</td>
<td>6.773</td>
<td>6.299</td>
<td>7.719</td>
</tr>
<tr>
<td></td>
<td>50</td>
<td>236.022</td>
<td>157.798</td>
<td>351.724</td>
<td>6.221</td>
<td>5.639</td>
<td>7.546</td>
</tr>
<tr>
<td>3</td>
<td>20</td>
<td>40.717</td>
<td>0.365</td>
<td>141.991</td>
<td>9.913</td>
<td>9.233</td>
<td>11.015</td>
</tr>
<tr>
<td></td>
<td>30</td>
<td>93.066</td>
<td>37.342</td>
<td>239.965</td>
<td>8.751</td>
<td>7.958</td>
<td>9.237</td>
</tr>
<tr>
<td></td>
<td>40</td>
<td>205.304</td>
<td>10.130</td>
<td>616.964</td>
<td>7.431</td>
<td>7.214</td>
<td>7.579</td>
</tr>
<tr>
<td></td>
<td>50</td>
<td>841.691</td>
<td>103.863</td>
<td>3379.935</td>
<td>7.272</td>
<td>6.732</td>
<td>8.044</td>
</tr>
<tr>
<td>4</td>
<td>20</td>
<td>60.077</td>
<td>1.353</td>
<td>181.429</td>
<td>10.855</td>
<td>10.350</td>
<td>11.389</td>
</tr>
<tr>
<td></td>
<td>30</td>
<td>294.756</td>
<td>57.802</td>
<td>693.769</td>
<td>10.077</td>
<td>9.058</td>
<td>10.585</td>
</tr>
<tr>
<td></td>
<td>40</td>
<td>1792.053</td>
<td>129.210</td>
<td>3204.703</td>
<td>8.138</td>
<td>7.412</td>
<td>8.597</td>
</tr>
<tr>
<td></td>
<td>50</td>
<td>6849.018</td>
<td>1908.091</td>
<td>10041.090</td>
<td>8.499</td>
<td>7.826</td>
<td>10.862</td>
</tr>
<tr>
<td>5</td>
<td>20</td>
<td>71.852</td>
<td>5.657</td>
<td>217.256</td>
<td>12.942</td>
<td>12.199</td>
<td>14.130</td>
</tr>
<tr>
<td></td>
<td>30</td>
<td>6255.652</td>
<td>2272.214</td>
<td>10016.737</td>
<td>11.346</td>
<td>10.969</td>
<td>11.632</td>
</tr>
<tr>
<td></td>
<td>50</td>
<td>20070.693</td>
<td>20013.265</td>
<td>20162.207</td>
<td>9.424</td>
<td>8.617</td>
<td>9.976</td>
</tr>
</tbody>
</table>

we have ensured that all the subproblems are convex. Table 2 suggests that we can solve large scale problems with relatively short times using the BB algorithm. This shows the efficiency of the BB algorithm in such a case. However, for $\alpha = 0.1$, some of the subproblems may be non-convex. The computational time of the BB algorithm significantly increases especially when $K \geq 4$ and $d \geq 40$. It takes more time for BARON to solve a nonconvex problem to optimality than SeDuMi to solve a convex problem to optimality especially when $K$ and $d$ become large. Therefore, the two algorithms have different merits and can be complementary to each other. In practical applications, it is often better to run BB starting from the nonlinear optimization algorithm. That is, we use the nonlinear optimization algorithm to find a good local optimal solution, and then use BB to improve upon the local solution.

We also compare the performances of the BB algorithm and SAA in solving Problem (16a)-(16b). Similar to Pagnoncelli et al. (2009), we reformulate the SAA of Problem (6a)-(6b) as the following mixed integer program (MIP)

\[
\begin{align*}
\text{minimize} & \quad h(x) \\
\text{subject to} & \quad f_0(x) + f(x)^\top \xi_i \leq Mz_i, \quad i = 1, \ldots, N, \\
& \quad \sum_{i=1}^N z_i \leq N\alpha, \\
& \quad x \in X, \quad z_i \in \{0, 1\}, \quad i = 1, \ldots, N.
\end{align*}
\]
where \( M = \max_{x \in X, i \in \{1, \ldots, N\}} f_0(x) + f(x)^\top \xi_i \) with \( \{\xi_i\}_{i=1}^N \) generated from the GMM. For the reformulation, the number of auxiliary 0-1 decision variables and the number of constraints are proportional to the sample size. We set \( h(x) \) to be a convex quadratic function \( x^\top H x + c^\top x \) and \( X \) to be \([-5, 5]^d\). We randomly generate the coefficients of the objective function \( h(x) \) as follows: (i) The elements of \( c \) are generated from the \([-100, 0]\) uniform distribution independently; (ii) The matrix \( H \) is generated from \( A A^\top \) where \( A \) is a \( d \times d \) matrix of independent random numbers. We set \( f_0(x) = -150, -100 \) corresponding to \( \alpha = 0.05, 0.1 \) respectively. The remaining parameter settings of Problem (16a)-(16b) and the BB algorithm are the same as above. We use CPLEX 12.8 to solve Problem (17a)-(17d) with parameters “mip.tolerances.mipgap” and “mip.tolerances.absmipgap” of CPLEX being \( 10^{-3} \) so that the optimization will terminate when the absolute or relative MIP gap tolerance is reached. In addition, as long as the computational time of the BB algorithm and SAA is larger than 3600s after an iteration, we terminate its implementation.

For SAA, setting the sample size is a challenging problem. Since we consider random distributions whose support is unbounded, the guidelines in Luedtke and Ahmed (2008) may not be used. In this paper, we adopt some appropriate sample sizes and study their properties. Specifically for \( \alpha = 0.05 \), the sample size \( N \) is set to be 100, 3000, 5000 and 8000, corresponding to \( d = 50, 80, 100 \) and 200 respectively, and for \( \alpha = 0.1 \), \( N \) is set to be 100, 300, 500 and 2000, corresponding to \( d = 20, 30, 40 \) and 50 respectively. To evaluate the performance of the SAA under the given sample size, we repeatedly implement the SAA five times under each combination of \( K \) and \( d \). In Tables 5 and 7, we report the optimal values derived by the two procedures under \( \alpha = 0.05 \) and \( \alpha = 0.1 \). The column \( h^* \) represents the optimal values obtained by the BB algorithm and the three columns “Ave”, “Min” and “Max” represent the average, minimal and maximal optimal values among the five runs. If the SAA fails to return a feasible solution of Problem (16a)-(16b), we discard that solution when reporting the results in Tables 5 and 7. In Tables 6 and 8, we report the computational time for both algorithms to reach the termination criterion. We denote by “N” for the case where the BB algorithm or the SAA fails to obtain a feasible solution of Problem (16a)-(16b) within the time limit.

From the experiments, we can observe that the SAA has some stability issue, which coincides with the findings in the literature (e.g., Pagnoncelli et al. (2009) and Peña-Ordieres et al. (2019)). In Table 5, there are a number of combinations (with “N”) where the SAA did not find feasible solutions for the five replications. For other rows where there are values, we may also encounter infeasibility for some of the replications. This suggests that the study on the sample size determination for SAA is important. For the case where \( \alpha = 0.05 \), Tables 5 and 6 suggest that when the dimension \( d \) is high, e.g., \( d = 100, 200 \), and the component number \( K \) is not very large, e.g., \( K = 2, 3, 4 \), the BB algorithm can often obtain smaller objective function values than the SAA within relatively shorter time. For the case where \( \alpha = 0.1 \), SAA still admits some instability in our experiments but the issue is much less severe. Table 7 suggests that the performance of the BB algorithm and SAA is close to each other in terms of the optimality. However, Table 8 shows that the computational time of the BB algorithm
Table 5: Optimal values obtained by the BB algorithm and the SAA with $\alpha = 0.05$

<table>
<thead>
<tr>
<th>$K$</th>
<th>$d$</th>
<th>$h^*$</th>
<th>BB Algorithm</th>
<th>SAA</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td></td>
<td></td>
<td>Ave</td>
<td>Min</td>
</tr>
<tr>
<td>2</td>
<td>50</td>
<td>-4010.422</td>
<td>-4010.130</td>
<td>-4010.422</td>
</tr>
<tr>
<td></td>
<td>80</td>
<td>-4975.532</td>
<td>N</td>
<td>N</td>
</tr>
<tr>
<td></td>
<td>100</td>
<td>-6023.129</td>
<td>-6023.059</td>
<td>-6023.059</td>
</tr>
<tr>
<td></td>
<td>200</td>
<td>-8952.419</td>
<td>N</td>
<td>N</td>
</tr>
<tr>
<td>3</td>
<td>50</td>
<td>-4530.938</td>
<td>-4530.107</td>
<td>-4530.938</td>
</tr>
<tr>
<td></td>
<td>80</td>
<td>-5671.678</td>
<td>-5669.218</td>
<td>-5670.473</td>
</tr>
<tr>
<td></td>
<td>100</td>
<td>-4786.794</td>
<td>N</td>
<td>N</td>
</tr>
<tr>
<td></td>
<td>200</td>
<td>-9582.504</td>
<td>-8914.516</td>
<td>-9003.624</td>
</tr>
<tr>
<td>4</td>
<td>50</td>
<td>-3331.731</td>
<td>-3331.731</td>
<td>-3331.731</td>
</tr>
<tr>
<td></td>
<td>80</td>
<td>-4418.467</td>
<td>N</td>
<td>N</td>
</tr>
<tr>
<td></td>
<td>100</td>
<td>-6082.795</td>
<td>-6031.317</td>
<td>-6031.317</td>
</tr>
<tr>
<td></td>
<td>200</td>
<td>-8110.376</td>
<td>-7324.050</td>
<td>-7324.050</td>
</tr>
<tr>
<td>5</td>
<td>50</td>
<td>-4527.809</td>
<td>-4527.361</td>
<td>-4527.809</td>
</tr>
<tr>
<td></td>
<td>80</td>
<td>-5011.244</td>
<td>-5011.244</td>
<td>-5011.244</td>
</tr>
<tr>
<td></td>
<td>100</td>
<td>-5295.994</td>
<td>-5290.826</td>
<td>-5290.826</td>
</tr>
<tr>
<td></td>
<td>200</td>
<td>N</td>
<td>-7728.917</td>
<td>-7747.468</td>
</tr>
</tbody>
</table>

Table 6: Computational time of the BB algorithm and the SAA with $\alpha = 0.05$

<table>
<thead>
<tr>
<th>$K$</th>
<th>$d$</th>
<th>Time (s)</th>
<th>BB Algorithm</th>
<th>SAA</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td></td>
<td>Ave (s)</td>
<td>Min (s)</td>
<td>Max (s)</td>
</tr>
<tr>
<td>2</td>
<td>50</td>
<td>1.608</td>
<td>0.162</td>
<td>0.133</td>
</tr>
<tr>
<td></td>
<td>80</td>
<td>3.839</td>
<td>38.378</td>
<td>10.836</td>
</tr>
<tr>
<td></td>
<td>100</td>
<td>1.572</td>
<td>2564.332</td>
<td>905.784</td>
</tr>
<tr>
<td></td>
<td>200</td>
<td>16.086</td>
<td>3624.730</td>
<td>3613.804</td>
</tr>
<tr>
<td>3</td>
<td>50</td>
<td>0.109</td>
<td>0.192</td>
<td>0.121</td>
</tr>
<tr>
<td></td>
<td>80</td>
<td>6.120</td>
<td>16.971</td>
<td>4.537</td>
</tr>
<tr>
<td></td>
<td>100</td>
<td>41.506</td>
<td>3612.926</td>
<td>3610.090</td>
</tr>
<tr>
<td></td>
<td>200</td>
<td>332.157</td>
<td>3619.292</td>
<td>3613.676</td>
</tr>
<tr>
<td>4</td>
<td>50</td>
<td>0.068</td>
<td>0.143</td>
<td>0.111</td>
</tr>
<tr>
<td></td>
<td>80</td>
<td>48.054</td>
<td>17.265</td>
<td>11.073</td>
</tr>
<tr>
<td></td>
<td>100</td>
<td>824.759</td>
<td>3605.084</td>
<td>3601.586</td>
</tr>
<tr>
<td></td>
<td>200</td>
<td>3472.138</td>
<td>3616.635</td>
<td>3608.167</td>
</tr>
<tr>
<td>5</td>
<td>50</td>
<td>0.065</td>
<td>0.183</td>
<td>0.114</td>
</tr>
<tr>
<td></td>
<td>80</td>
<td>0.092</td>
<td>5.436</td>
<td>4.886</td>
</tr>
<tr>
<td></td>
<td>100</td>
<td>403.830</td>
<td>805.552</td>
<td>66.034</td>
</tr>
<tr>
<td></td>
<td>200</td>
<td>3600.398</td>
<td>3619.643</td>
<td>3614.901</td>
</tr>
</tbody>
</table>
Table 7: Optimal values obtained by the BB algorithm and the SAA with $\alpha = 0.1$

<table>
<thead>
<tr>
<th>$K$</th>
<th>$d$</th>
<th>$h^*$</th>
<th>BB Algorithm Ave</th>
<th>BB Algorithm Min</th>
<th>BB Algorithm Max</th>
<th>SAA Ave</th>
<th>SAA Min</th>
<th>SAA Max</th>
</tr>
</thead>
<tbody>
<tr>
<td>2</td>
<td>20</td>
<td>-1948.010</td>
<td>-1948.010</td>
<td>-1948.010</td>
<td>-1948.010</td>
<td>30</td>
<td>-3021.558</td>
<td>-3021.558</td>
</tr>
<tr>
<td></td>
<td>30</td>
<td>-3021.558</td>
<td>-3021.558</td>
<td>-3021.558</td>
<td>-3021.558</td>
<td>40</td>
<td>-3673.854</td>
<td>-3674.635</td>
</tr>
<tr>
<td></td>
<td>40</td>
<td>-3673.676</td>
<td>-3674.635</td>
<td>-3673.096</td>
<td>-3673.096</td>
<td>50</td>
<td>-4311.580</td>
<td>-4317.489</td>
</tr>
<tr>
<td>3</td>
<td>20</td>
<td>-1878.896</td>
<td>-1878.840</td>
<td>-1878.896</td>
<td>-1878.616</td>
<td>30</td>
<td>-2351.701</td>
<td>-2351.319</td>
</tr>
<tr>
<td></td>
<td>30</td>
<td>-2351.701</td>
<td>-2351.319</td>
<td>-2350.664</td>
<td>-2350.664</td>
<td>40</td>
<td>-2869.429</td>
<td>-2868.893</td>
</tr>
<tr>
<td></td>
<td>40</td>
<td>-2868.893</td>
<td>-2868.893</td>
<td>-2867.026</td>
<td>-2867.026</td>
<td>50</td>
<td>N</td>
<td>N</td>
</tr>
<tr>
<td>4</td>
<td>20</td>
<td>-2162.301</td>
<td>-2162.065</td>
<td>-2162.301</td>
<td>-2161.260</td>
<td>30</td>
<td>-2551.568</td>
<td>-2551.568</td>
</tr>
<tr>
<td></td>
<td>30</td>
<td>-2551.568</td>
<td>-2551.568</td>
<td>-2551.568</td>
<td>-2551.568</td>
<td>40</td>
<td>-2752.266</td>
<td>-2752.266</td>
</tr>
<tr>
<td></td>
<td>40</td>
<td>-2751.507</td>
<td>-2752.266</td>
<td>-2750.591</td>
<td>-2750.591</td>
<td>50</td>
<td>-3576.113</td>
<td>-3576.113</td>
</tr>
<tr>
<td>5</td>
<td>20</td>
<td>-1724.862</td>
<td>-1724.541</td>
<td>-1724.862</td>
<td>-1724.041</td>
<td>30</td>
<td>-3002.829</td>
<td>-3002.779</td>
</tr>
<tr>
<td></td>
<td>40</td>
<td>-3932.573</td>
<td>-3932.573</td>
<td>-3931.193</td>
<td>-3931.193</td>
<td>50</td>
<td>N</td>
<td>N</td>
</tr>
</tbody>
</table>

Table 8: Computational time of the BB algorithm and the SAA with $\alpha = 0.1$

<table>
<thead>
<tr>
<th>$K$</th>
<th>$d$</th>
<th>Time (s)</th>
<th>BB Algorithm Ave (s)</th>
<th>BB Algorithm Min (s)</th>
<th>BB Algorithm Max (s)</th>
<th>SAA Ave (s)</th>
<th>SAA Min (s)</th>
<th>SAA Max (s)</th>
</tr>
</thead>
<tbody>
<tr>
<td>2</td>
<td>20</td>
<td>10.347</td>
<td>0.346</td>
<td>0.134</td>
<td>0.997</td>
<td>30</td>
<td>27.913</td>
<td>0.232</td>
</tr>
<tr>
<td></td>
<td>30</td>
<td>27.913</td>
<td>0.232</td>
<td>0.168</td>
<td>0.329</td>
<td>40</td>
<td>322.739</td>
<td>1.664</td>
</tr>
<tr>
<td></td>
<td>40</td>
<td>322.739</td>
<td>1.664</td>
<td>0.873</td>
<td>2.394</td>
<td>50</td>
<td>1528.294</td>
<td>20.711</td>
</tr>
<tr>
<td>3</td>
<td>20</td>
<td>23.344</td>
<td>0.213</td>
<td>0.165</td>
<td>0.267</td>
<td>30</td>
<td>36.402</td>
<td>0.195</td>
</tr>
<tr>
<td></td>
<td>30</td>
<td>36.402</td>
<td>0.195</td>
<td>0.154</td>
<td>0.232</td>
<td>40</td>
<td>99.499</td>
<td>0.811</td>
</tr>
<tr>
<td></td>
<td>40</td>
<td>99.499</td>
<td>0.811</td>
<td>0.542</td>
<td>1.365</td>
<td>50</td>
<td>3633.176</td>
<td>182.171</td>
</tr>
<tr>
<td>4</td>
<td>20</td>
<td>16.699</td>
<td>0.131</td>
<td>0.122</td>
<td>0.142</td>
<td>30</td>
<td>32.944</td>
<td>0.151</td>
</tr>
<tr>
<td></td>
<td>30</td>
<td>32.944</td>
<td>0.151</td>
<td>0.148</td>
<td>0.159</td>
<td>40</td>
<td>115.089</td>
<td>0.307</td>
</tr>
<tr>
<td></td>
<td>40</td>
<td>115.089</td>
<td>0.307</td>
<td>0.252</td>
<td>0.455</td>
<td>50</td>
<td>225.416</td>
<td>5.640</td>
</tr>
<tr>
<td>5</td>
<td>20</td>
<td>23.220</td>
<td>0.170</td>
<td>0.115</td>
<td>0.232</td>
<td>30</td>
<td>40.889</td>
<td>0.291</td>
</tr>
<tr>
<td></td>
<td>30</td>
<td>40.889</td>
<td>0.291</td>
<td>0.265</td>
<td>0.334</td>
<td>40</td>
<td>120.371</td>
<td>1.106</td>
</tr>
<tr>
<td></td>
<td>40</td>
<td>120.371</td>
<td>1.106</td>
<td>0.560</td>
<td>1.352</td>
<td>50</td>
<td>3690.349</td>
<td>375.484</td>
</tr>
</tbody>
</table>
is much longer than that of the SAA especially when both $d$ and $K$ become large. This may be due to that with $\alpha = 0.1$ some of the subproblems (15a)-(15c) become nonconvex, and thus are more difficult to solve. The computational results suggest that the BB algorithm may have competitiveness in the stability. It may also enjoy a computational efficiency when the confidence level is small.

Furthermore, we show that the BB algorithm can also handle Problem (16a)-(16b) with a more general convex objective function. Note that this problem may not be handled by SAA. Specifically, we set $h(x) = x^\top H x + \sum_{i=1}^{d} \exp(5 - x_i)$. Here, we let $f_0(x) = -50$, $d = 30$, $\alpha = 0.01$ and $K = 3$, and solve all the subproblems (15a)-(15c) using BARON. The remaining input parameters are the same as the ones above. The performances of the BB algorithm within four replications are shown in Figure 2 where the relative error is defined as $|U - L|/|L|$. When the BB algorithm has not found feasible solutions until the current iteration, we let the relative error be 1. Figure 2 suggests that throughout the four replications, the BB algorithm can solve the problem to optimality quickly since the relative error will reduce to 0 after a few iterations.

![Figure 2: Performances of the BB algorithm within four replications for Problem (16a)-(16b) with $h(x) = x^\top H x + \sum_{i=1}^{d} \exp(5 - x_i)$.](image)

26
4.2 A Portfolio Optimization Problem

In this subsection, we consider a portfolio optimization problem with the following form:

\[
\begin{align*}
\text{maximize} \quad & \mathbb{E}[r^\top x] \\
\text{subject to} \quad & \text{VaR}_{1-\alpha, P_x}(-r^\top x) \leq v, \\
& 1^\top x = 1,
\end{align*}
\]

where the portfolio consists of \(d\) assets whose return rates are \(d\)-dimensional random vector \(r = (r_1, \cdots, r_d)^T\) following a distribution \(P_x\), \(x\) is the \(d\)-dimensional decision vector consisting of the percentile proportions of each asset, and \(\text{VaR}_{1-\alpha}(\cdot)\) denotes the VaR of a random variable at confidence level \(1 - \alpha\). The total wealth is set to 1. In addition, we prohibit the short selling of assets by restricting \(x\) to \(\mathbb{R}_+^d\). The objective of this problem is to maximize the expected total return rate of the portfolio while controlling the risk (VaR of the portfolio) below a certain threshold \(v\). It is clear that Problem (18a)-(18c) can be reformulated as the following CCP:

\[
\begin{align*}
\text{maximize} \quad & \mathbb{E}[r^\top x] \\
\text{subject to} \quad & \text{Pr}_{r \sim P_x}\{-r^\top x \leq v\} \geq 1 - \alpha, \\
& 1^\top x = 1.
\end{align*}
\]

4.2.1 Constructing the Portfolio

In our experiments, we construct a hedge fund portfolio consisting of 12 indices of Dow Jones Credit Suisse Broad Hedge Fund, which are listed in Table 13 of Appendix A.3. Therefore, the dimension \(d\) is set to 12. We collect 273 monthly returns of these 12 indices from April 1994 to December 2016, and use the data to model the distribution \(P_x\). We first report some descriptive statistics of the collected data in Table 9. From Table 9 we can see that the skewness of the

<table>
<thead>
<tr>
<th>Asset</th>
<th>Mean(%)</th>
<th>Standard deviation(%)</th>
<th>Skewness</th>
<th>Kurtosis</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>0.57</td>
<td>1.85</td>
<td>-2.66</td>
<td>20.51</td>
</tr>
<tr>
<td>2</td>
<td>-0.41</td>
<td>4.69</td>
<td>0.73</td>
<td>4.52</td>
</tr>
<tr>
<td>3</td>
<td>0.62</td>
<td>3.87</td>
<td>-0.84</td>
<td>9.59</td>
</tr>
<tr>
<td>4</td>
<td>0.67</td>
<td>1.76</td>
<td>-2.07</td>
<td>12.67</td>
</tr>
<tr>
<td>5</td>
<td>0.76</td>
<td>1.78</td>
<td>-2.10</td>
<td>14.34</td>
</tr>
<tr>
<td>6</td>
<td>0.63</td>
<td>1.91</td>
<td>-1.62</td>
<td>9.69</td>
</tr>
<tr>
<td>7</td>
<td>0.48</td>
<td>1.14</td>
<td>-0.90</td>
<td>7.47</td>
</tr>
<tr>
<td>8</td>
<td>0.44</td>
<td>1.49</td>
<td>-4.80</td>
<td>40.29</td>
</tr>
<tr>
<td>9</td>
<td>0.88</td>
<td>2.54</td>
<td>0.18</td>
<td>7.98</td>
</tr>
<tr>
<td>10</td>
<td>0.74</td>
<td>2.64</td>
<td>0.02</td>
<td>6.94</td>
</tr>
<tr>
<td>11</td>
<td>0.44</td>
<td>3.35</td>
<td>0.05</td>
<td>2.83</td>
</tr>
<tr>
<td>12</td>
<td>0.63</td>
<td>1.43</td>
<td>-1.72</td>
<td>9.79</td>
</tr>
</tbody>
</table>
majority of the return rates is negative, which is significantly different from a zero skewness of the normal distribution. Meanwhile, most of the asset return rates have kurtosis greater than three, indicating that the underlying distributions of the monthly return rates are more outlier-prone than the normal distribution.

Therefore, it is intuitively not appropriate to fit the distribution \( P_* \) to Gaussian distributions. In the experiments, we use some GMM to fit the data. We use the EM algorithm to estimate the parameters of the GMM after specifying the number of components. Because too many components may lead to the badly conditioned estimated component covariances, we restrict the component number to certain range while setting the regularization value to \( 10^{-4} \). In order to avoid inaccurate parameter estimations caused by initial values required by the EM algorithm, we design a two-stage method to fit a GMM to the underlying data as follows.

**Stage 1.** For each number of mixture components, fit multiple GMMs to the underlying data by repeatedly launching the EM algorithm with various starting points, and choose the GMM with the largest loglikelihood.

**Stage 2.** Compare the Akaike Information Criterion (AIC) (see, e.g., Bishop 2006) among various number of mixture components, and choose the number of mixture components for which the fitted GMM is with the least AIC value.

In the first stage, we launch the EM algorithm 500 times. For each run, the EM algorithm terminates when either the maximum iteration reaches \( 10^4 \) or the loglikelihood function value meets termination tolerance \( 10^{-6} \). We use the K-means++ algorithm for the initialization of the EM algorithm. Since K-means++ algorithm randomly selects a sample from our data as the first initial component center, the resulting starting points for the EM algorithm vary with each run. Here we only consider the number of components ranging from 1 to 10. Following the above two-stage method, we choose three components for the best fitting GMM with the least AIC statistics value as shown in Table 10.

<table>
<thead>
<tr>
<th>Number</th>
<th>1</th>
<th>2</th>
<th>3</th>
<th>4</th>
<th>5</th>
</tr>
</thead>
<tbody>
<tr>
<td>AIC</td>
<td>-17567.54</td>
<td>-17856.76</td>
<td>-17919.76</td>
<td>-17852.36</td>
<td>-17780.03</td>
</tr>
<tr>
<td>Number</td>
<td>6</td>
<td>7</td>
<td>8</td>
<td>9</td>
<td>10</td>
</tr>
<tr>
<td>AIC</td>
<td>-17667.90</td>
<td>-17569.00</td>
<td>-17437.54</td>
<td>-17324.80</td>
<td>-17196.77</td>
</tr>
</tbody>
</table>

We further analyze some of the marginal distributions. We consider three hedge funds that have the largest absolute values of skewness and kurtosis, i.e., Convertible arbitrage, Event driven distressed and Fixed income arbitrage. We consider the three marginal distributions for the fitted Gaussian distribution and GMM. Figure 3 displays the corresponding quantile plots for the fitted Gaussian distribution, where the horizontal axis is the sample quantile based on a set of sample with sample size 1000 simulated from the Gaussian distribution, and the vertical axis is the empirical quantile of the real data. In contrast, Figure 4 displays the quantile...
Figure 3: Quantile plots of the hedge fund return rates under Gaussian distribution: (a) Convertible arbitrage; (b) Event driven distressed; (c) Fixed income arbitrage

Figure 4: Quantile plots of the hedge fund return rates under Gaussian mixture model: (a) Convertible arbitrage; (b) Event driven distressed; (c) Fixed income arbitrage

plots for the fitted GMM. Note that in the quantile plots, we have eliminated the highest 2.5% and lowest 2.5% of both real data and simulated data so that the extreme returns can be avoided. From the plots we can see that GMM provides a superior fitting than the Gaussian distribution. Finally, we remark that using GMM to fit the financial data has been advocated in the literature, see, e.g., Buckley et al. (2008).

4.2.2 Computational Results

With the distribution specified, we now implement the nonlinear optimization algorithm and the BB algorithm to handle the problem. Similarly, we let $q(x) = \text{VaR}_{1-\alpha,P}(-r^T x) - v$. It can be verified that $\nabla q(x)$ has the following closed form:

$$
\nabla q(x) = \sum_{j=1}^{K} \pi_j \phi \left( \frac{q(x) + x^T \mu_j}{\sqrt{x^T \Sigma_j x}} \right)^{-1} \sum_{j=1}^{K} \pi_j \phi \left( \frac{q(x) + x^T \mu_j}{\sqrt{x^T \Sigma_j x}} \right) \left( -\frac{\mu_j}{\sqrt{x^T \Sigma_j x}} + \frac{(q(x) + x^T \mu_j) \Sigma_j x}{(x^T \Sigma_j x)^{3/2}} \right).
$$
We embed $\nabla q(x)$ in fmincon to solve Problem (18a)-(18c). The parameters of the fmincon function are the same as the ones in Section 4.1.

In the BB algorithm, we use the solutions returned by the nonlinear optimization algorithm as the initial upper bound in the BB algorithm and set the relative termination tolerance to be $10^{-2}$. In addition, we set “DeltaA” in BARON to $10^{-4}$. The remaining parameter settings are the same as for the test problem above.

Figure 5 shows the optimal values of Problem (18a)-(18c) obtained by both algorithms under different combinations of $v$ and $\alpha$. For $\alpha = 0.01$, the value of $v$ ranges from 2% to 4% with step being 0.05%; for $\alpha = 0.05$, the value of $v$ ranges from 1% to 3% with step being 0.05%; for $\alpha = 0.1$, the value of $v$ ranges from 0.5% to 2.5% with step being 0.05%. From Figure 5, we can find that for various combinations of $v$ and $\alpha$, the nonlinear optimization algorithm based on quantile reformulation can indeed find the optimal solutions, and at the same time, the BB algorithm verifies the global optimality.

Finally, we briefly compare the performance of the optimal portfolio derived from the GMM and the Gaussian distribution respectively. We vary the values of $v$ (The values are the same as in Figure 5.) and obtain the corresponding optimal solutions of Problem (19a)-(19c) for these two input models. We then calculate the historical mean return rate and historical VaR corresponding to the optimal solutions. The result is shown in Figure 6, where the vertical axis and the horizontal axis represent the historical mean return rate and historical VaR respectively. For each value of $v$, a “◦” or a “×” is plotted in the figure. Note that if the results for different $v$ values coincide, then the “◦” or the “×” also coincide, which means the numbers of “◦” or “×” may appear different in the figure. From Figure 6, we find that when $\alpha = 0.01$, using the GMM we can obtain a better portfolio frontier. Furthermore, we can find that the empirical VaR has a larger deviation from the imposed constraints for the Gaussian distribution setting. This may suggest that the GMM shows some superiority than the Gaussian distribution for
Figure 6: Historical risk-return pairs under different settings: GMM versus Gaussian distribution when (a) $\alpha = 0.01$, (b) $\alpha = 0.05$ and (c) $\alpha = 0.1$.

capturing the tail risks and for the portfolio selection.

4.2.3 Out-of-Sample Analysis

We further compare the out-of-sample performances of the optimal portfolios generated from the GMM and the Gaussian distribution fitted by the same empirical return rate data. We divide the whole data set into the in-sample and out-of-sample parts. More specifically, the initial in-sample data set consists of the monthly returns of the 12 indices from April 1994 to December 2011 with the remaining part being the initial out-of-sample data. The in-sample data are used to estimate the parameters of the GMM and Gaussian distribution with the EM algorithm. The out-of-sample data are used to compare the performances of different portfolios constructed by solving the Problem (18a)-(18c) with the fitted GMM and the Gaussian distribution. More specifically, we conduct the out-of-sample analysis of the constructed portfolios using a back testing strategy by calculating performance criteria of different portfolios with the monthly returns in out-of-sample period. We gradually increase the sample size of the in-sample data set with the most recent three monthly return rate data and repeatedly solve Problem (18a)-(18c) every three months to generate the monthly portfolio rebalance strategies based on the newly fitted GMM and Gaussian distribution.

Figures 7 and 8 illustrate the evolution of the values of different portfolios constructed by the GMM and Gaussian distribution under different combinations of $v$ and $\alpha$ using the monthly rebalancing strategy. As shown in the figures, given the same initial portfolio value 100, the optimal portfolios derived from the GMM can often achieve better portfolio values than using a Gaussian distribution in the long run. The GMM based model can performs well in the case of $v = 0.01$, which may suggest that it is more suitable for extreme cases than the Gaussian model. Note that when $v = 0.01$ and $\alpha = 0.05$, there are no feasible portfolios under the Gaussian distribution. However, from the figures we find that the volatility of the optimal portfolios derived from the GMM is larger than that from the Gaussian distribution. Tables 11 and 12 show respectively the out-of-sample statistical results of the optimal portfolios generated by the

31
Figure 7: Evolution of portfolio values during out-of-sample period for $\alpha = 0.05$ under different settings: GMM versus Gaussian distribution when (a) $v = 0.01$, (b) $v = 0.015$ and (c) $v = 0.02$.

Figure 8: Evolution of portfolio values during out-of-sample period for $\alpha = 0.1$ under different settings: GMM versus Gaussian distribution when (a) $v = 0.01$, (b) $v = 0.015$ and (c) $v = 0.02$.

GMM and Gaussian distribution, including the final portfolio value (Finval), average return rate ($R_{\text{Ave}}$), maximum return rate ($R_{\text{Max}}$), minimum return rate ($R_{\text{Min}}$), standard deviation (Std), and the ratio of average return rate over standard deviation ($R_{\text{Ave}}$/Std) of monthly return rate. In accordance with the evolution of the portfolio values in Figures 7 and 8, Tables 11 and 12 suggest that the portfolios derived from the GMM often show a better performance than that from the Gaussian distribution in terms of the final portfolio values and the average return rate under various combinations of $v$ and $\alpha$. However, as reflected in Figures 7 and 8, the standard deviations in Tables 11 and 12 show that the volatility of portfolios generated by the GMM is larger than that by the Gaussian distribution under all combinations of $v$ and $\alpha$. Thus, in terms of the ratio of average return rate over standard deviation, GMM does not enjoy a superiority over the Gaussian distribution.

5 Conclusions

In this paper, we studied using GMM to model the randomness in CCPs. We analyzed the structures and properties of CCPs with GMM. Based on the results we studied several scenarios and discussed how to handle the CCPs in these scenarios for efficient computation. We also conducted some numerical experiments to demonstrate the implementation of our approach. We expect that the approach proposed in this paper could generalize the scope and applicability
Table 11: Statistical results of portfolio values and return rates for the Gaussian distribution during the out-of-sample period

<table>
<thead>
<tr>
<th>$\alpha$</th>
<th>$v$</th>
<th>Finval</th>
<th>$R_{Ave}$</th>
<th>$R_{Min}$</th>
<th>$R_{Max}$</th>
<th>Std</th>
<th>$R_{Ave}/Std$</th>
</tr>
</thead>
<tbody>
<tr>
<td>0.05</td>
<td>0.01</td>
<td>100.000</td>
<td>N</td>
<td>N</td>
<td>N</td>
<td>N</td>
<td>N</td>
</tr>
<tr>
<td>0.015</td>
<td></td>
<td>164.823</td>
<td>0.851</td>
<td>-6.768</td>
<td>3.676</td>
<td>1.734</td>
<td>0.491</td>
</tr>
<tr>
<td>0.02</td>
<td></td>
<td>175.422</td>
<td>0.981</td>
<td>-11.505</td>
<td>6.256</td>
<td>2.812</td>
<td>0.349</td>
</tr>
<tr>
<td>0.1</td>
<td>0.01</td>
<td>163.873</td>
<td>0.840</td>
<td>-6.377</td>
<td>3.486</td>
<td>1.666</td>
<td>0.505</td>
</tr>
<tr>
<td>0.015</td>
<td></td>
<td>175.531</td>
<td>0.990</td>
<td>-12.697</td>
<td>6.839</td>
<td>3.092</td>
<td>0.320</td>
</tr>
<tr>
<td>0.02</td>
<td></td>
<td>170.550</td>
<td>0.981</td>
<td>-16.924</td>
<td>9.216</td>
<td>4.174</td>
<td>0.235</td>
</tr>
</tbody>
</table>

Table 12: Statistical results of portfolio values and return rates for the GMM during the out-of-sample period

<table>
<thead>
<tr>
<th>$\alpha$</th>
<th>$v$</th>
<th>Finval</th>
<th>$R_{Ave}$</th>
<th>$R_{Min}$</th>
<th>$R_{Max}$</th>
<th>Std</th>
<th>$R_{Ave}/Std$</th>
</tr>
</thead>
<tbody>
<tr>
<td>0.05</td>
<td>0.01</td>
<td>133.622</td>
<td>0.657</td>
<td>-5.096</td>
<td>3.261</td>
<td>1.490</td>
<td>0.441</td>
</tr>
<tr>
<td>0.015</td>
<td></td>
<td>178.169</td>
<td>0.991</td>
<td>-6.849</td>
<td>5.059</td>
<td>2.179</td>
<td>0.455</td>
</tr>
<tr>
<td>0.02</td>
<td></td>
<td>180.437</td>
<td>1.049</td>
<td>-13.147</td>
<td>8.377</td>
<td>3.485</td>
<td>0.301</td>
</tr>
<tr>
<td>0.1</td>
<td>0.01</td>
<td>187.178</td>
<td>1.083</td>
<td>-8.206</td>
<td>6.389</td>
<td>2.583</td>
<td>0.419</td>
</tr>
<tr>
<td>0.015</td>
<td></td>
<td>177.446</td>
<td>1.044</td>
<td>-14.963</td>
<td>9.261</td>
<td>4.086</td>
<td>0.255</td>
</tr>
<tr>
<td>0.02</td>
<td></td>
<td>183.633</td>
<td>1.131</td>
<td>-18.668</td>
<td>10.600</td>
<td>4.740</td>
<td>0.239</td>
</tr>
</tbody>
</table>
of CCPs.

A Appendix

A.1 Gradient Formula of Henrion and Möller (2012)

Consider (5) and suppose that \( \xi \) follows \( \mathcal{N}(\mathbb{Z}|\mu, \Sigma) \). Let

\[
S^{(i)} = \Sigma - \frac{1}{b_i^\top \Sigma b_i} \Sigma b_i b_i^\top \Sigma, \quad w^{(i)} = \mu + \frac{x_i - b_i^\top \mu}{b_i^\top \Sigma b_i} \Sigma b_i, \quad i = 1, \cdots, m. \tag{20}
\]

From Lemma 3.1 of Henrion and Möller (2012), there exists a factorization

\[
S^{(i)} = L^{(i)} L^{(i)^\top}, \tag{21}
\]

where \( L^{(i)} \) is of dimension \( k \times (k-1) \) and rank \( L^{(i)} = k-1 \).

Lemma 1 (Theorem 3.3 of Henrion and Möller (2012)) Let \( x \) be such that \( Bz \leq x \) is nondegenerate. Suppose that \( \Sigma \) is positive definite. Then

\[
\frac{\partial}{\partial x_i} \Pr \{ B\xi \leq x \} = g_i(x_i) \Pr \{ B^{(i)} L^{(i)} \xi^{(i)} \leq x^{(i)} - B^{(i)} w^{(i)} \}, \tag{22}
\]

where \( \xi^{(i)} \sim \mathcal{N}(\mathbb{Z}|0, I_{k-1}) \), \( B^{(i)} \) results from \( B \) by deleting row \( i \), \( x^{(i)} \) results from \( x \) by deleting component \( i \), \( L^{(i)} \) and \( w^{(i)} \) are defined in (20)-(21), and \( g_i \) is the one-dimensional Gaussian density with mean \( b_i^\top \mu \) and variance \( b_i^\top \Sigma b_i \). Moreover, the inequality system \( B^{(i)} L^{(i)} y \leq x^{(i)} - B^{(i)} w^{(i)} \) is nondegenerate.

A.2 Proof of Proposition 5

Proof. Since the region \( \Delta \) in Problem (15a)-(15c) is a subset of the initial region

\[
\Delta^0 := \{ y \in \mathbb{R}^K \mid y^0_j \leq y_j \leq y^0_j, j = 1, \cdots, K \},
\]

we have \( y^0_j \leq y_j \), \( j = 1, \cdots, K \). If \( y^0_j \geq 1/2, j = 1, \cdots, K \), then \( y_j \geq 1/2, j = 1, \cdots, K \). Note that \( y^0_j = \max \{0, 1 - \alpha/\pi_j\} \). Then \( y^0_j \geq 1/2, j = 1, \cdots, K \) is equivalent to \( \pi_j \geq 2\alpha, j = 1, \cdots, K \). This concludes the proof. \( \square \)

A.3 Dow Jones Credit Suisse Broad Hedge Fund Indices

References


Table 13: Dow Jones Credit Suisse Broad Hedge Fund indices

1 Convertible arbitrage
2 Dedicated short bias
3 Emerging markets
4 Event driven
5 Event driven distressed
6 Event driven multistrategy
7 Event driven risk arbitrage
8 Fixed income arbitrage
9 Global macro
10 Long/short equity
11 Managed futures
12 Multistrategy


425-440.