Chance Constrained Programs with Mixture Distributions

Zhaolin Hu, Wenjie Sun
School of Economics and Management, Tongji University, Shanghai 200092, China

Shushang Zhu
Department of Finance and Investment, Sun Yat-Sen Business School, Sun Yat-Sen University, Guangzhou 510275, China

Abstract

Chance constrained programs (CCP) are important models in stochastic optimization. In the main conventional literature on CCPs, the underlying distribution that models the randomness of the problem is usually assumed to be given in advance. However, in practice, such a distribution needs to be specified by the modelers based on the data/information available. This is called input modeling. In this paper we consider input modeling in CCPs. We propose to use mixture distributions to fit the data available and to model the randomness. We demonstrate the merits of using mixture distributions and show how to handle the CCPs with mixture distributions. Furthermore, we consider several scenarios that arise from practical applications and analyze how the problem structures could embrace alternative optimization techniques. We also conduct numerical experiments to demonstrate our approach.

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 into a CCP. The literature on CCPs can be dated back to Charnes et al. (1958), who first considered a single CCP, and Miller and Wagner (1965), who first considered a joint CCP. 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), Hu et al. (2013), Hanasusanto et al. (2017) and references therein.

In the framework of SO, the randomness of the system is modeled by the random parameters. In the literature, standard approaches typically assume that the distribution of the underlying random vector is given in advance. For instance, we often assume the random vector follows a multivariate normal (or t) distribution. However, in practice, there is usually no distribution that is ready for use. Decision makers often need to specify the distribution based on the information available. This important process is called input modeling. Traditionally there is a gap between the

input modeling part and the optimization part. Analysis in the SO literature has mainly focused on the optimization part of the SO problem. For instance, to our knowledge, while there have been extensive studies on how to solve CCPs, the input modeling for CCPs has received relatively insufficient discussion. In this paper, we make an emphasis on the input modeling issue for SO: Since the optimal solutions of SO models critically depend on the input distributions, input modeling is a critical step of SO, and is perhaps equally important as the solution techniques.

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 some distribution to the data. When there are no data, the modeler often needs to specify the distribution based on some subjective judgement. In simulation, the data based approaches can roughly be divided into the parametric approach and the nonparametric approach. In the parametric approach, decision makers typically select a parametric distribution family, e.g., a multivariate normal distribution, and then use statistical methods to determine (usually estimate) the parameters of the distribution, e.g., the mean vector and the covariance matrix. In the nonparametric approach, decision makers do not assume the distribution belongs to any parametric family. Rather, they use some nonparametric methods to fit the distribution. For instance, the kernel density estimation methods, which are 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 often obtain data for the random parameters. Thus it is often beneficial to mining the data to specify the input distribution for SO. Such a data driven approach has been very natural and attractive. In this paper we propose such an input modeling approach for CCPs. In contrast to the parametric approach and the nonparametric approach, we propose to use mixture distributions (or called mixture models) to model the randomness of the parameters. 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. 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 distribution, which has been studied extensively in the literature, and has been used widely in various areas, including economics, finance, engineering and social sciences. Bringing in the mixture models in CCPs, we show that this approach has many merits and we demonstrate that the approach can in some sense extend the current research results for CCPs.

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 CCPs with mixture distributions can be transformed as a weighted sum of probability functions with component distributions of the mixture models. Based on the result, we analyze in details three important scenarios.

In the first scenario, 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. We call it a linear separable randomness and decision vector scenario. Many engineering management problems can be modeled as such CCPs. There have been much study on the problems. 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 Gaussian mixture model. In the second scenario, we consider a class of CCPs where the constraint within the probability function takes a product form of the randomness and the decision vector. We call it a linear product form scenario. This class of models have found applications in various areas. We show that for the Gaussian mixture distribution, the problems can essentially 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. Because the nonlinear optimization techniques typically could only guarantee stationary points. In this paper, we further propose a globally convergent branch-and-bound (BB) algorithm to solve this class of CCPs. Solving CCPs to global optimality is an important contribution of this paper. As the third scenario, we extend the second scenario and consider a class of mixed integer CCPs. In recent years, 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). Even with a simple linear product form, the class of mixed integer CCPs that we consider have been very challenging to solve. Due to the integer variables, it is often difficult to implement simulation based techniques to handle the problems. There are, however, some interesting results when the underlying distribution is Gaussian. In this paper, we analyze the problems under the context of Gaussian mixture models. We show the mixed integer CCPs with Gaussian mixture distributions can be transformed to some deterministic integer programs. The transformation provides a potential for implementing more optimization techniques to handle the difficult original problems.

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 three scenarios and demonstrate the potential wide applications of our approach. We conduct some preliminary numerical experiments in Section 4. Section 5 concludes the paper.

2 Mixture Distributions and Model Structures

2.1 Probability Function with Mixture Distribution

We start from the well known model chance constrained program (CCP):

$$\begin{aligned} \underset{x\in X}{\text{minimize}} \quad & h(x) && (1)\\ \text{subject to} \quad & \Pr_{\sim P_*}\{A(x,\xi)\} \geq 1-\alpha. && (2) \end{aligned}$$

In Problem (1), x is the decision vector, which belongs to a set $X \subset \Re^d$, $h : \Re^d \to \Re$ is a real-valued function which models the objective, ξ is a k-dimensional vector of random parameters and the support of ξ, denoted as Ξ, is a closed subset of $\Re^k$, $A(x,\xi)$ is a random event (driven by ξ) that depends on x, and the notation $\Pr_{\sim P_*}$ denotes that the probability is taken with respect to (w.r.t.) P_*, where P_* is usually referred to as the underlying distribution. Constraint (2) is called a chance constraint. A typical instance of $A(x,\xi)$ is the following

$$A(x,\xi) := c_1(x,\xi) \leq 0, \cdots, c_m(x,\xi) \leq 0,$$

where $c_i : \Re^d \times \Xi \to \Re, i = 1, \cdots, m$ are real-valued functions. In this setting, Problem (1) is called a single CCP if $m = 1$ and a joint CCP if $m > 1$. In such a scenario, constraint (2) requires that the m constraints are satisfied at least with a probability $1-\alpha$. 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 (1) comes from the chance constraint (2).

When Problem (1) is being studied, the underlying distribution P_* is typically assumed to be given. However, as discussed in Section 1, in practice, such a distribution can only be inferred. In this paper, we model that P_* is a mixture distribution that takes the following expression

$$p_*(z) = \sum_{j=1}^{K} \pi_j p_j(z), \tag{3}$$

where $p_j(z), j = 1, \cdots, K$ are probability distributions which are called component distributions, and $\pi_j > 0, j = 1, \cdots, K$ are mixing coefficients satisfying $\sum_{j=1}^{K} \pi_j = 1$. 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. In (3), p_* and p_j could be probability density functions or

probability mass functions. As noted, we also call (3) a mixture model. Mixture models have been studied for long in the area of statistics. For instance, McLachlan and Peel (2000) studied finite mixture models and built various interesting results for the models.

2.2 Gaussian Mixture Distribution

A most well known instance of the mixture models is the Gaussian mixture distribution. The Gaussian mixture model takes the following form

$$p_*(z) = \sum_{j=1}^{K} \pi_j \mathcal{N}(z|\,\mu_j, \Sigma_j), \tag{4}$$

where each component $\mathcal{N}(z|\,\mu_j, \Sigma_j)$ is a (multivariate) normal density function with mean vector μ_j and covariance matrix Σ_j. Gaussian mixture distribution is widely used in machine learning literature, see, e.g., Bishop (2009). Using Gaussian mixture models to approximate a distribution has been very popular in the literature. The approximation theory has also been studied, see, for instance, Zeevi and Meir (1997), Li and Barron (2000) and Maugis-Rabusseau and Michel (2013). Recently, Gaussian mixture models 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 random parameters in a CCP sometimes follow a continuous skewed distribution, an asymmetrical distribution, or a multimodal distribution. These distribution structures typically hide in the data that are collected. Consequently, conventional parametric families, e.g., a multivariate normal 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 ξ, denoted as $\xi_1, \xi_2, \cdots, \xi_N$. We can use the following EM algorithm (Chapter 9 of Bishop 2006) to obtain a Gaussian mixture distribution.

Expectation Maximization (EM)

Step 1. Initialize μ_j, Σ_j and π_j, $j = 1, \cdots, K$.

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

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

where

$$N_j = \sum_{n=1}^{N} \gamma_{nj}.$$

Step 4. Check whether the convergence criterion is satisfied. If not, return to Step 2.

It is well known that such EM algorithm will converge to some local maxima of the log likelihood function. The EM algorithm has been imbedded in some softwares. For instance, in Matlab, we can directly call the EM algorithm package to conduct the estimation for the Gaussian mixture model.

2.3 Model Structures

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. Readers are referred to, e.g., Nemirovski and Shapiro (2006), Chen et al. (2010), and Hong et al. (2011) for the discussions. There are two main streams of methodologies for tackling CCPs. One stream is to use a deterministic optimization approach, which has been explored extensively in the optimization literature, see, e.g., Nemirovski and Shapiro (2006) and Chen et al. (2010). In this approach, some deterministic optimization model is often constructed to somehow approximate the CCP. The deterministic model is then solved and the solution is used to approximate that of the original problem. The other stream is to use a simulation approach. In this approach, a set of sample are typically simulated from the underlying distribution and based on them some optimization procedures are designed to solve the sample approximation problem, see, e.g., Pagnoncelli et al. (2009) and Hong et al. (2011).

The simulation approach is typically 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. Therefore, in the literature, the CCPs are usually assumed to have certain structures. In this paper, we explore certain structures and properties of Constraint (2), and use them to design efficient procedures to solve Problem (1) under different scenarios. As a simple but important result, we have the following theorem.

Theorem 1. *Suppose the distribution P_* takes the form of (3). Then*

$$\Pr_{\sim P_*}\{A(x,\xi)\} = \sum_{j=1}^{K} \pi_j \Pr_{\sim P_j}\{A(x,\xi)\}.$$

Proof. Let $\mathbb{1}_{\{A(x,\xi)\}}$ denote the indicator function, which equals 1 if $A(x,\xi)$ happens and 0 otherwise. Then

$$\Pr_{\sim P_*}\{A(x,\xi)\} = \mathrm{E}_{P_*}\left[\mathbb{1}_{\{A(x,\xi)\}}\right] = \sum_{j=1}^{K} \pi_j \mathrm{E}_{P_j}\left[\mathbb{1}_{\{A(x,\xi)\}}\right] = \sum_{j=1}^{K} \pi_j \Pr_{\sim P_j}\{A(x,\xi)\}.$$

This concludes the proof of the theorem. □

Theorem 1 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. For instance, when the distributions $p_j, j = 1, \cdots, K$ are normal, sometimes more efficient methods, than a crude simulation method, may be used to compute (or estimate) the probability function values.

Problem (1) 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 general 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. 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 assume that the probability function in (2) is smooth. We analyze the gradient estimation for the chance constraint (2). Since the probability function with a mixture distribution can be separated, as stated in Theorem 1, the gradient can also be separated directly. From Theorem 1 we immediately have the following theorem.

Theorem 2. *Suppose that the distribution P_* takes the form of (3). Then*

$$\nabla_x \Pr_{\sim P_*}\{A(x,\xi)\} = \sum_{j=1}^{K} \pi_j \nabla_x \Pr_{\sim P_j}\{A(x,\xi)\},$$

given that the probability functions involved are differentiable.

Theorem 2 shows that the gradient of the probability function with a mixture distribution is simply 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 Theorem 2 guarantees that we can handle the gradient for the mixture model efficiently. In next section we will give concrete examples that support the statement.

Theorem 2 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 Theorem 2 to simplify the problem. Of course, in other probability optimization problems, e.g., optimizing a probability function, Theorem 2 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 Linear Separable Randomness and Decision Vector

We first consider a class of CCPs that have very simple structures. The chance constraint in the models has the following expression

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

where B is a matrix of appropriate dimension. Note that the constraint above is indeed a joint chance constraint. In the chance constraint the random vector ξ 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 the chance constraint above and implemented the models 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 for this case. As their main contribution, they built that under the Gaussian distribution setting, calculations of the gradients can be reduced to calculations of the probability function values.

For a general distribution, it is not known how to derive similar results. However, based on Theorem 2 we can obtain a similar gradient formula for the probability function with the Gaussian mixture distribution. To ease the analysis, we define

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

Theorem 3.3 (or Theorem 4.1) of Henrion and Möller (2012) built an expression for the gradient of $F^j(x)$. We thus take the gradient expression $\nabla_x F^j(x)$ as given. Based on Theorem 2, we can derive an expression for the gradient of $F^*(x)$. We summarize the result in the following proposition.

Proposition 1. *Suppose P_* is a Gaussian mixture distribution defined by (4). Let x be such that $By \leq x$ is nondegenerate in the sense of Henrion and Möller (2012). Then*

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

Proposition 1 can be viewed as a generalization of the gradient formulas of Henrion and Möller (2012). Considering that Gaussian mixture distributions can often approximate many distributions very well, such a generalization is very useful. Consider now the following CCP

$$\begin{array}{ll} \underset{x \in X}{\text{minimize}} & h(x) \\ \text{subject to} & \Pr_{\sim P_*} \{B\xi \leq x\} \geq 1 - \alpha. \end{array}$$

We can use some nonlinear optimization procedure as in Henrion and Möller (2012) to solve the problem. During the procedure, we can implement Proposition 1 to estimate the function value and the gradient for the chance constraint.

Proposition 1 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 Theorem 2 to access the corresponding gradients and to conduct optimization accordingly.

3.2 Continuous Linear Product Form

We next consider a class of CCPs with a linear product structure. Suppose that $f_i(x), i = 0, 1, \cdots, k$ are real valued functions defined on X. Let $f(x) = (f_1(x), \cdots, f_k(x))^\top$. We consider the following CCP

$$\begin{array}{ll} \underset{x \in X}{\text{minimize}} & h(x) \qquad (5) \\ \text{subject to} & \Pr_{\sim P_*} \left\{ f_0(x) + f(x)^\top \xi \leq 0 \right\} \geq 1 - \alpha. \qquad (6) \end{array}$$

In contrast to the separable structure considered in preceding subsection, the random vector ξ and the decision vector x in the chance constraint (6) take a product form. This class of chance constraints have 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). It is typically difficult to derive some deterministic representation for the chance constraint (6). One exception is when the underlying distribution P_* falls in the normal distribution family.

Suppose that the distribution P_* follows a multivariate normal distribution $\mathcal{N}(z|\, \mu, \Sigma)$. Then the chance constraint (6) is equivalent to the following constraint

$$\Phi^{-1}(1-\alpha)\sqrt{f(x)^\top \Sigma f(x)} + \left(f_0(x) + f(x)^\top \mu \right) \leq 0, \qquad (7)$$

where $\Phi^{-1}(\cdot)$ is the inverse of the standard normal distribution function. As it is easy to evaluate $\Phi^{-1}(\cdot)$, (7) is essentially a deterministic constraint. Suppose further that $h(x)$, $f_0(x)$ and $f(x)$ are linear in x and $\alpha \leq 0.5$. Then the constraint (7) is known as a second-order cone constraint. In this setting, Problem (5) 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 consider the Gaussian mixture distribution and conduct some analysis for such a setting. Suppose that the distribution P_* takes the form of (3). Then from Theorem 1 we have the constraint (6) is the same as

$$\sum_{j=1}^{K} \pi_j \mathrm{Pr}_{\sim P_j} \left\{ f_0(x) + f(x)^\top \xi \leq 0 \right\} \geq 1 - \alpha.$$

Note that P_j is $\mathcal{N}(z|\, \mu_j, \Sigma_j)$. The constraint above can be further written as

$$\sum_{j=1}^{K} \pi_j \Phi \left(\frac{-\left[f_0(x) + f(x)^\top \mu_j \right]}{\sqrt{f(x)^\top \Sigma_j f(x)}} \right) \geq 1 - \alpha \tag{8}$$

where $\Phi(\cdot)$ is the standard normal distribution function. The function $\Phi(\cdot)$ can essentially be viewed as a deterministic function since we can evaluate its function values easily. Furthermore, its gradient (derivative) is simply the density function of the standard normal distribution which is known explicitly. Therefore, we can transform Problem (5) to a deterministic optimization problem, for which the function values and gradients of the constraint can be evaluated efficiently. The transformation then allows us to use conventional nonlinear optimization algorithms to solve the problem.

Due to the non-convexity of CCPs, nonlinear optimization procedures typically could guarantee stationary points for the CCPs. However, the structure of (8) offers some potential for searching the global optimal solutions for the CCPs. By introducing an auxiliary decision vector $y = (y_1, \cdots, y_K)^\top$, we can transform the chance constraint in Problem (5) equivalently to

$$\mathrm{Pr}_{\sim P_j} \left\{ f_0(x) + f(x)^\top \xi \leq 0 \right\} \geq y_j, \ \sum_{j=1}^{K} \pi_j y_j \geq 1 - \alpha.$$

Then based on the analysis above, we can obtain the following equivalent formulation of Problem (5):

$$\begin{aligned} \underset{x \in X, y}{\text{minimize}} \quad & h(x) \\ \text{subject to} \quad & \Phi^{-1}(y_j) \sqrt{f(x)^\top \Sigma_j f(x)} + \left(f_0(x) + f(x)^\top \mu_j \right) \leq 0, j = 1, \cdots, K, \\ & \sum_{j=1}^{K} \pi_j y_j \geq 1 - \alpha, 0 \leq y_j \leq 1, j = 1, \cdots, K. \end{aligned} \tag{9}$$

The equivalence here guarantees that if (x^*, y^*) is an optimal solution of Problem (9), then x^* is optimal to Problem (5). Problem (9) 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. We will pursue this direction in Section 3.4.

3.3 Mixed Integer Linear Product Form

In Problem (5), the decision variables are continuous. We next consider a class of CCPs that have the same structure with (5) but part of the decision variables are integer-valued. The models could be expressed as follows

$$\begin{array}{ll} \underset{x\in X}{\text{minimize}} & h(x) \\ \text{subject to} & \Pr_{\sim P_*}\left\{f_0(x) + f(x)^\top \xi \le 0\right\} \ge 1-\alpha, \\ & x \in Z^{d_1} \times \Re^{d_2}, \end{array} \tag{10}$$

where Z is the set of integers and $d_1 + d_2 = d$. The model requires that the first d_1 elements of x are integer-valued and the rest d_2 elements are continuous. Of course, one can replace Z^{d_1} with $\{0,1\}^{d_1}$, i.e., the first n_1 elements of x are 0-1 valued. Such structured models are called mixed integer CCPs. Many real world problems, e.g., facility locations, may be modeled as mixed integer CCPs. However, the problems are typically very difficult to solve. One possible simulation approach is to use the sample average approximation. In this approach, one simulates a sample from the distribution P_* and then uses the sample mean function to approximate the probability function to obtain a sample counterpart. By introducing an auxiliary 0-1 decision variable for each scenario in the sample, one then reformulates the sample counterpart as a mixed integer program. However, for this approach, the number of auxiliary 0-1 decision variables is proportional to the sample size. Thus when the sample size is large, the problem becomes very challenging to solve.

Suppose now P_* takes the form of (4). As discussed in preceding subsection, the constraints in Problem (10) can be transformed to (8) with $x \in Z^{d_1} \times \Re^{d_2}$. Along this direction, we obtain some deterministic constraint with mixed integer variables. Compared to the original mixed integer chance constraint, the mixed integer deterministic constraint becomes slightly easier to handle. It is possible to implement some mixed integer optimization algorithms or heuristic algorithms to obtain some good solutions. Note that for the setting, one tractable case is when $K = 1$, i.e., the underlying distribution P_* degenerates to a multivariate normal distribution. With this setting, as discussed in preceding subsection, we can further transform (10) into a mixed integer deterministic program which has the constraint (7). In particular, when $h(x)$, $f_0(x)$ and $f(x)$ are all linear in x and $\alpha \le 0.5$. The resulting problem is a mixed integer conic program (MICP). MICP has been

studied widely in recent years, see, e.g., Atamtürk et al. (2012). It can be solved by using the BB type procedures, usually imbedded in optimization softwares such as CPLEX.

As discussed in preceding subsection, when P_* takes the form of (4), Problem (10) can be formulated equivalently to the following problem

$$\begin{array}{ll} \underset{x\in X, y}{\text{minimize}} & h(x) \\ \text{subject to} & \Phi^{-1}(y_j)\sqrt{f(x)^\top \Sigma_j f(x)} + \Big(f_0(x) + f(x)^\top \mu_j\Big) \le 0, j = 1, \cdots, K, \\ & \sum_{j=1}^{K} \pi_j y_j \ge 1-\alpha, \\ & x \in Z^{d_1} \times \Re^{d_2}. \end{array} \tag{11}$$

Problem (11) shares similar structures as Problem (9). Indeed, Problem (9) can be regarded as a special case ($d_1 = 0$) of Problem (11). In general, Problem (11) is more difficult than Problem (9) due to that some elements of x are integer variables.

3.4 A Branch-and-Bound Algorithm

In this subsection, according to the special structure of the mixture model, we propose a branch-and-bound (BB) type procedure to handle Problem (11). Denote

$$\Delta := \{y \in \Re^K \mid \underline{y}_j \le y_j \le \overline{y}_j, j = 1, \ldots, K\}$$

as the rectangle of y. Notice that $\Phi^{-1}(\cdot)$ is a monotonically increasing function. We now consider the following relaxation of Problem (11) over Δ:

$$\begin{array}{ll} \underset{x\in X, y\in\Delta}{\text{minimize}} & h(x) \\ \text{subject to} & \Phi^{-1}(\underline{y}_j)\sqrt{f(x)^\top \Sigma_j f(x)} + \Big(f_0(x) + f(x)^\top \mu_j\Big) \le 0, j = 1, \cdots, K, \\ & \sum_{j=1}^{K} \pi_j y_j \ge 1-\alpha, 0 \le y_j \le 1, j = 1, \cdots, K, x \in Z^{d_1} \times \Re^{d_2}. \end{array} \tag{12}$$

At each iteration of our BB method, the domain of the vector y is partitioned into sub-rectangles. Solving the subproblem of (12) on a sub-rectangle of y provides a lower bound for the optimal value of the subproblem of (11) on the same sub-rectangle. If the optimal solution is also feasible to (11), then its objective value provides an upper bound for the optimal value of (11). At each iteration, a new subproblem of (12) on certain sub-rectangle is solved and a better solution could possibly be identified during the solution process. For any subproblem of (12) that is infeasible or its optimal value is greater than or equal to the best attained objective value of (11), the corresponding sub-rectangle is cut 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 considerations. This process repeats until all the sub-rectangles are removed. The method is now formally described as follows.

Algorithm 1

Step 0 (**Initialization**): Set $i := 0$, $\Delta^i := \{y \in \Re^K \mid 0 = \underline{y}_j \leq y_j \leq \overline{y}_j = 1, j = 1, \ldots, K\}$, $v^* := \infty$, $\underline{v}^i := -\infty$, $\Omega := \{\Delta^i, \underline{v}^i\}$, $flag := 0$.

Step 1:

IF $\Omega \neq \emptyset$, **GOTO** Step 2;

ELSE IF $flag = 0$, Problem (11) is infeasible, **STOP**;

ELSE, (x^*, y^*) is an optimal solution to Problem (11), **STOP**.

Step 2: Choose and remove instance $\{\Delta^i, \underline{v}^i\}$ from Ω with the smallest $\underline{v}^i$, solve Problem (12) with $y \in \Delta^i$.

IF it is infeasible, **GOTO** Step 1;

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

Step 3:

IF $v^i \geq v^*$, **GOTO** Step 1;

ELSE IF $v^i < v^*$ and (x^i, y^i) is not a feasible solution to Problem (11), **GOTO** Step 4;

ELSE, $v^* := v^i$, $(x^*, y^*) := (x^i, y^i)$, $flag := 1$, delete from Ω all instances $\{\Delta^i, \underline{v}^i\}$ with $\underline{v}^i \geq v^*$, **GOTO** Step 1.

Step 4: Subdivide Δ^i into two rectangles Δ^i_1 and Δ^i_2 by dividing the longest edge of Δ^i at its midpoint, set $\underline{v}^i_1 := \underline{v}^i_2 := v^i$, $\Omega := \Omega \bigcup \{\Delta^i_1, \underline{v}^i_1\} \bigcup \{\Delta^i_2, \underline{v}^i_2\}$, **GOTO** Step 2.

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 4, we have $int\{\Delta^i_1\} \bigcap int\{\Delta^i_2\} = \emptyset$ and $\Delta^i = \Delta^i_1 \bigcup \Delta^i_2$, where $int(S)$ denotes the interior of set S.

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 (11) or verifies the infeasibility of Problem (11). The following proposition gives the global convergent property of Algorithm 1.

Proposition 2. *Suppose that the vector-valued function $f(x)$ is continuous and bounded on X. If Algorithm 1 generates an infinite sequence $\{x^i, y^i\}_{i\in\mathcal{N}}$, then any limit point (x^*, y^*) of this sequence satisfying $y_j^* \in (\delta, 1-\delta), j = 1, \cdots K$ with any given $0 < \delta < 0.5$ is an optimal solution to Problem (11).*

Proof. Denote

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

Suppose $\{x^{i_h}, y^{i_h}\}_{i_h\in\mathcal{N}}$ is a convergent subsequence of $\{x^i, y^i\}_{i\in\mathcal{N}}$ and $\{\Delta^{i_h}\}_{i_h\in\mathcal{N}}$ is the corresponding nested sequence of rectangles, respectively. Denote (x^*, y^*) as the the limit point of $\{x^{i_h}, y^{i_h}\}_{i_h\in\mathcal{N}}$, and assume that $y_j^* \in (\delta, 1-\delta)$ for a given $0 < \delta < 0.5$. Note that $x^* \in Z^{d_1} \times \Re^{d_2}$. Since we always divide the longest edge of Δ^i in Step 4 of Algorithm 1, we have $\lim_{i_h\to\infty}\left(\left(\overline{y}_j^{i_h}\right) - \left(\underline{y}_j^{i_h}\right)\right) = 0$. Thus for some sufficiently large N, both $\underline{y}_j^{i_h}$ and $\overline{y}_j^{i_h}$ lie in $(\delta, 1-\delta)$ for any $i_h > N$. Noting that $\Phi^{-1}(\cdot)$ is a monotonically increasing and continuous function over $(\delta, 1-\delta)$, together with the facts that $F_j\left(x^{i_h}, \underline{y}_j^{i_h}\right) \le 0$ and $f(x)$ is continuous and bounded on X, we get

$$\begin{aligned}
F_j\left(x^*, y_j^*\right) &= \lim_{i_h\to\infty} F_j\left(x^{i_h}, y_j^{i_h}\right) \\
&\le \lim_{i_h\to\infty}\left(F_j\left(x^{i_h}, y_j^{i_h}\right) - F_j\left(x^{i_h}, \underline{y}_j^{i_h}\right)\right) \\
&= \lim_{i_h\to\infty}\left(\Phi^{-1}\left(y_j^{i_h}\right) - \Phi^{-1}\left(\underline{y}_j^{i_h}\right)\right)\sqrt{f\left(x^{i_h}\right)^\top \Sigma_j f\left(x^{i_h}\right)} \\
&\le \lim_{i_h\to\infty}\left(\Phi^{-1}\left(\overline{y}_j^{i_h}\right) - \Phi^{-1}\left(\underline{y}_j^{i_h}\right)\right)\sqrt{f\left(x^{i_h}\right)^\top \Sigma_j f\left(x^{i_h}\right)} \\
&= 0,
\end{aligned}$$

which implies that (x^*, y^*) is a feasible solution to Problem (11). Denote the optimal value of Problem (11) by v_{opt}. Notice that, in Algorithm 1, we always choose $\{\Delta^i, \underline{v}^i\}$ with the smallest $\underline{v}^i$. Thus $\{\underline{v}^{i_h} : \underline{v}^{i_h} = h(x^{i_h})\}_{i_h\in\mathcal{N}}$ is a nondecreasing sequence bounded from above by v_{opt}. We thus have

$$h(x^*) = \lim_{i_h\to\infty} h(x^{i_h}) \le v_{opt},$$

which implies that (x^*, y^*) is an optimal solution to Problem (11). □

In our numerical implementation of the proposed BB algorithm, we stop further branching if both

$$|v^* - \min_i\{\underline{v}^i\}| \le \varepsilon \text{ and } \max_j\{F_j(x, y_j)\} \le \varepsilon$$

are satisfied, where ε is a given sufficiently small tolerance parameter for generating an approximate optimal solution.

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 certain procedures to solve the problems. In this paper, we focus on the linear case which has been studied widely in the literature and management practice. In this case, the functions f_0 and f are all linear in x. Then Problem (12) has a conic program structure. Specifically, for $d_1 = 0$, if $\underline{y}_j \geq 1/2$ for $j = 1, \cdots, K$, Problem (12) is an SOCP. We can use conventional softwares, e.g., SeDuMi (Sturm (1999)), to solve it efficiently. In the more general case, we can transform Problem (12) to a mixed integer quadratically constrained quadratic program. In the latter case, we can use some commercial softwares, e.g., BARON (see, e.g., Tawarmalani and Sahinidis (2005) and Sahinidis (2017)), to solve the problem for global optimality. Finally, we remark that for Problem (11), the number of branch variables, which is the number of mixture components, is usually very small. Thus the BB method can work efficiently.

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 value-at-risk (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 R2016b and run on an Intel Core i7 PC with 3.40 GHz and 16 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 for global optimality.

4.1 A Test Problem

We consider a specific instance of the CCP (5). As discussed in Section 3.2, Problem (5) can be transformed as

$$\begin{array}{ll} \underset{x \in X}{\text{minimize}} & h(x) \\ \text{subject to} & \displaystyle\sum_{j=1}^{K} \pi_j \Phi\left(\frac{-[f_0(x) + f(x)^\top \mu_j]}{\sqrt{f(x)^\top \Sigma_j f(x)}} \right) \geq 1 - \alpha. \end{array} \tag{13}$$

In this specific instance, we set $h(x)$ to be a linear function $c^\top x$, $f_0(x)$ to be a constant function, and $f(x) = (f_1(x), \ldots, f_k(x))^\top$ to be a linear function vector $a + B^\top x$ where a is a constant vector and B is a $k \times d$ matrix. In the experiment, we set the component number $K = 2$. We randomly generate the coefficients of the objective function $h(x)$ and the parameters of the Gaussian mixture distribution as follows: (i) the elements of c are generated from the $[-10, 0]$ uniform distribution independently; (ii) the elements of μ_j $(j = 1, \cdots, K)$ are respectively generated from the $[0, 5]$ uniform distribution independently; (iii) Σ_j $(j = 1, \cdots, K)$ are derived from UU^T where the elements

of U independently follow the $[0,1]$ uniform distribution; (iv) $\pi_j = r_j / \left(\sum_{i=1}^{K} r_i\right)$ $(j = 1, \cdots, K)$ where r_j is generated independently from the $[0,1]$ uniform distribution. For each combination of d and α we randomly generate the input parameters above five times. For the simulated data, we use both the nonlinear optimization algorithm (Matlab optimization function fmincon) and the BB algorithm (Algorithm 1) to solve Problem (13) respectively.

For the nonlinear optimization approach, we need the function values and the gradients of the chance constraint. For ease of analysis, we let $g(x) = \Pr_{\sim P_*}\left\{f_0(x) + f(x)^\top \xi \leq 0\right\}$. It is easy to verify that the gradient of $g(x)$ has the following closed form

$$\nabla g(x) = \sum_{j=1}^{K} \pi_j \varphi\left(\frac{-[f_0(x) + f(x)^\top \mu_j]}{\sqrt{f(x)^\top \Sigma_j f(x)}}\right)\left(\frac{f(x)^\top \mu_j B^\top \Sigma_j f(x) - B^\top \mu_j f(x)^\top \Sigma_j f(x)}{\sqrt[3]{f(x)^\top \Sigma_j f(x)}}\right),$$

where $\varphi(x)$ refers to the density function of the standard normal distribution. Then, we can embed the gradient $\nabla g(x)$ in the nonlinear optimization algorithm (Matlab 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^{-7}.

To implement the BB algorithm, we use the local optimal solutions or the infeasible ones obtained by the nonlinear optimization algorithm as starting points. The algorithm is terminated if the largest difference between the upper bound and the current optimal value is within a tolerance level ε. In the experiments we set ε to be 10^{-5}. In order to speed up the BB algorithm, we prune the subproblems for which $||\overline{y} - \underline{y}|| \leq 10^{-3}$.

The optimal values derived by the two procedures are listed in Table 1. In Table 1, "N" denotes the failure of the fmincon function to find any feasible solution to Problem (13), with the corresponding change percentage denoted as 1000% when the BB algorithm successfully obtains feasible solutions. As can be seen from Table 1, there are four columns for each combination of α and d. Column 3 (Change (%)) refers to the relative improvement percentage of the optimal value obtained by the BB algorithm with respect to the one obtained by the fmincon function. Column 4 (CPUT(s)) reports the CPU time (seconds) for the BB algorithm to terminate. These results suggest that the BB algorithm spends reasonable computational time in improving the solutions obtained by the fmincon function.

Furthermore, we briefly illustrate the performance of the BB algorithm in solving mixed integer counterpart of Problem (13) (i.e., we impose the constraint $x \in Z^{d_1} \times \Re^{d_2}$). We set $d_1 = 2$, $d_2 = 8$ and $\alpha = 0.3$. The remaining input parameters are the same as the ones above. The performances of the BB algorithm for the four replications are shown in Figure 1. Figure 1 suggests that for the four replications, the BB algorithm can not only quickly find feasible solutions to the mixed integer counterpart of Problem (13) but also constantly improve the optimal solutions found so far. However, it takes the BB algorithm more CPU time to terminate for the mixed integer counterpart

Table 1: Performances of the nonlinear optimization algorithm (fmincon) and the BB algorithm for Problem (13)

	α=0.1				α=0.2			
d	fmincon	BB	Change(%)	CPUT(s)	fmincon	BB	Change(%)	CPUT(s)
10	-39.06	-43.78	10.78	4.35	-42.14	-42.98	2.00	3.87
10	N	-41.90	1000	38.03	-20.11	-58.58	191.21	15.76
10	N	-51.27	1000	21.20	N	-54.21	1000	57.05
10	-57.10	-59.18	3.52	12.98	-83.68	-84.99	1.57	54.54
10	-49.33	-50.92	3.11	9.91	-57.24	-58.40	2.03	4.88
20	N	-41.09	1000	33.68	-41.95	-48.63	15.91	59.80
20	-45.06	-46.67	3.45	26.64	N	-54.72	1000	32.89
20	-35.44	-36.65	3.31	43.74	-48.67	-66.17	35.97	21.34
20	-45.31	-49.19	7.89	115.86	-56.46	-57.89	2.53	19.47
20	N	-37.12	1000	26.25	-7.06	-45.12	539.56	28.50
30	N	-38.77	1000	28.49	N	-58.22	1000	44.93
30	N	-30.68	1000	28.61	-46.25	-48.57	5.02	85.47
30	N	-32.77	1000	34.68	-38.72	-48.96	26.42	58.17
30	N	-30.13	1000	157.14	-46.53	-54.12	16.29	25.47
30	N	-35.98	1000	26.57	N	-48.11	1000	49.69
	α=0.3				α=0.4			
d	fmincon	BB	Change(%)	CPUT(s)	fmincon	BB	Change(%)	CPUT(s)
10	N	-39.86	1000	19.02	-50.10	-69.28	27.69	276.71
10	-43.02	-67.71	36.47	4.41	-31.14	-45.87	32.12	39.20
10	-46.41	-70.20	33.89	10.59	-79.88	-80.19	0.39	10.42
10	-53.56	-56.27	4.82	42.70	-102.83	-108.86	5.53	188.32
10	-61.33	-75.07	18.31	4.95	-47.29	-47.32	0.06	80.50
20	-73.48	-74.95	1.96	40.84	N	-73.06	1000	43.49
20	-82.51	-83.93	1.69	54.68	-61.09	-95.96	36.34	287.49
20	-62.58	-66.88	6.42	499.94	-69.93	-83.82	16.56	109.81
20	-27.35	-59.58	54.09	48.33	N	-65.85	1000	24.54
20	-107.76	-109.62	1.70	49.70	-34.92	-77.68	55.05	212.35
30	-43.55	-50.51	13.77	133.15	-69.76	-97.20	28.23	25.49
30	-55.87	-61.61	9.31	67.83	-52.88	-82.53	35.92	145.63
30	N	-70.25	1000	29.98	-64.23	-80.11	19.82	23.95
30	-91.19	-94.89	3.90	136.63	-59.91	-110.45	45.76	495.37
30	-16.19	-72.96	77.82	103.30	-64.24	-100.33	35.97	89.73

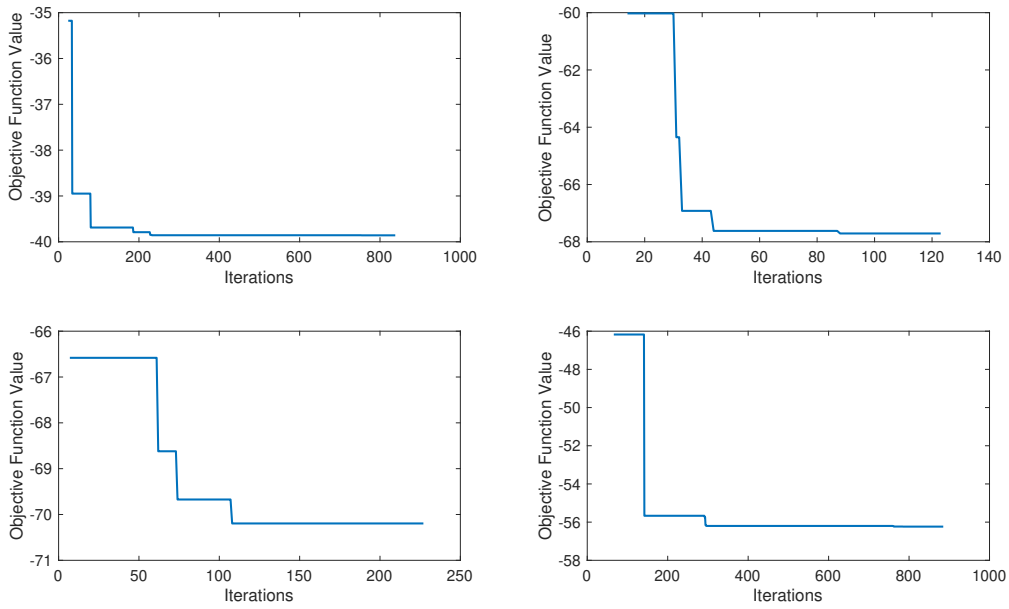

Figure 1: Performances of the BB algorithm for the four replications for the mixed integer counterpart of Problem (13)

than for Problem (13). The CPU times (seconds) for the four replications are 182.61, 26.25, 52.14 and 187.36 respectively.

4.2 A Portfolio Optimization Problem

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

$$\begin{aligned} \underset{x\in\Re^d_+}{\text{maximize}} \quad & \mathrm{E}[r^\top x] \\ \text{subject to} \quad & \mathrm{VaR}_{1-\alpha}(-r^\top x) \le v, \\ & \mathbf{1}^\top x = 1, \end{aligned} \tag{14}$$

where the portfolio consists of d assets whose return rates are d-dimensional random vector r following a distribution P_*, x is the d-dimensional decision vector consisting of the percentile proportions of each asset, and $\mathrm{VaR}_{1-\alpha}(\cdot)$ 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 $\Re^d_+$. The objective of this problem is to maximize the expected total return rate of the portfolio while controlling the risk (the VaR of the portfolio) below a certain threshold v. It is clear that Problem

(14) can be reformulated as the following CCP:

$$\begin{aligned} \underset{x\in\Re^d_+}{\text{maximize}} \quad & \mathrm{E}[r^\top x] \\ \text{subject to} \quad & \Pr_{\sim P_*}\{-r^\top x \le v\} \ge 1-\alpha, \\ & \mathbf{1}^\top x = 1. \end{aligned} \tag{15}$$

4.2.1 Constructing the Portfolio

In our experiments, we construct a hedge fund portfolio consisting of twelve Dow Jones Credit Suisse Broad Hedge Fund indices, which are listed in Table 2. Therefore, the dimension d is set to

Table 2: 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

12. We collect 273 monthly returns of these 12 indices from March 1994 to December 2016, and use the data to model the distribution P_*.

We first report some descriptive statistics of the collected data in Table 3. From Table 3 we can see that the skewness of the majority of the return rates is negative, which is significantly different from a zero skewness of the normal distribution. Moreover, 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 for us to fit the distribution P_* to Gaussian distributions. In the experiments, we use some Gaussian mixture distribution to fit the data. We use the EM algorithm to estimate the parameters of the Gaussian mixture distribution 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 Gaussian mixture distribution to the underlying data as follows.

Stage 1. For each number of mixture components, fit multiple Gaussian mixture distributions to

Table 3: Descriptive statistics of monthly return rate of hedge fund indices

Asset	Mean(%)	Standard deviation(%)	Skewness	Kurtosis
1	0.57	1.85	-2.66	20.51
2	-0.41	4.69	0.73	4.52
3	0.62	3.87	-0.84	9.59
4	0.67	1.76	-2.07	12.67
5	0.76	1.78	-2.10	14.34
6	0.63	1.91	-1.62	9.69
7	0.48	1.14	-0.90	7.47
8	0.44	1.49	-4.80	40.29
9	0.88	2.54	0.18	7.98
10	0.74	2.64	0.02	6.94
11	0.44	3.35	0.05	2.83
12	0.63	1.43	-1.72	9.79

the underlying data by repeatedly launching the EM algorithm with various starting points, and choose the Gaussian mixture distribution 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 Gaussian mixture distribution is with the least AIC value.

Following the above two-stage method, we choose three components for the best fitting Gaussian mixture distribution with the least AIC statistics value as shown in Table 4.

Table 4: Comparison of AIC statistics over varying number of components

Number	2	3	4	5	6	7	8
AIC	-17856.76	-17919.76	-17852.36	-17780.03	-17667.90	-17569.00	-17437.54

4.2.2 Computational Results

With the distribution specified, we now implement the nonlinear optimization algorithm and the BB algorithm to handle the problem. Following Section 3.2, we can transform Problem (15) as

$$\begin{aligned} \underset{x\in\Re^d_+}{\text{maximize}} \quad & \sum_{j=1}^{K} \pi_j \mu_j^\top x \\ \text{subject to} \quad & \sum_{j=1}^{K} \pi_j \Phi\left(\frac{v+\mu_j^\top x}{\sqrt{x^\top \Sigma_j x}}\right) \geq 1-\alpha, \\ & \mathbf{1}^\top x = 1. \end{aligned} \tag{16}$$

Similarly, we let $g(x) = \Pr_{\sim P_*}\{-r^\top x \le v\}$. It can be verified that $\nabla g(x)$ has the following closed form:

$$\nabla g(x) = \sum_{j=1}^{K} \pi_j \phi\left(\frac{v + \mu_j^\top x}{\sqrt{x^\top \Sigma_j x}}\right)\left(-\frac{\mu_j^\top \Sigma_j x - x^\top \Sigma_j x \mu_j}{\sqrt[3]{x^\top \Sigma_j x}}\right)$$

where $\phi(x)$ denotes the density function of the standard normal distribution. We embed $\nabla g(x)$ in the nonlinear optimization algorithm (fmincon) to solve the problem. The parameters of the fmincon function are the same as the ones in subsection 4.1.

As in the previous example, in the BB algorithm, we use the solutions returned by the nonlinear optimization algorithm as starting points. The parameter setting is the same as for the test problem above. We consider different combinations of v and α. Figures 2 and 3 summarize the computational results for these combinations. Figure 2 includes the optimal values of Problem (16) obtained by

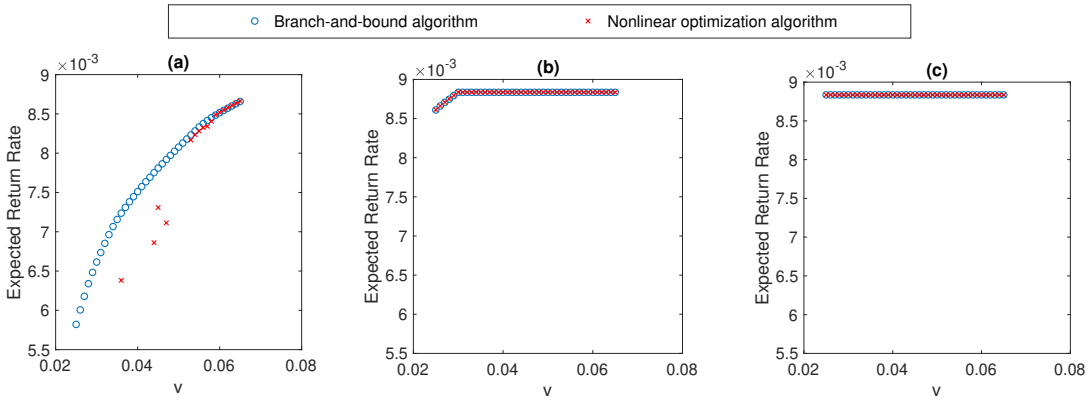

Figure 2: Risk-return pairs obtained by the BB algorithm and the nonlinear optimization algorithm for Problem (16) when (a) $\alpha = 0.01$, (b) $\alpha = 0.05$ and (c) $\alpha = 0.1$. For each fixed α, the value of v ranges from 2.5% to 6.5% with step being 0.1%.

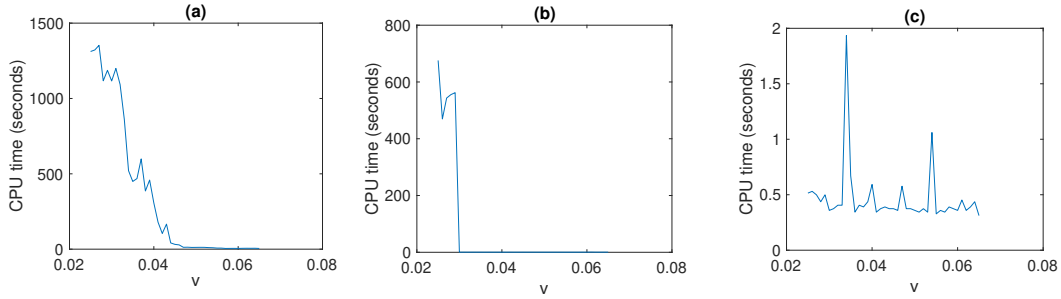

Figure 3: CPU time (seconds) needed by the BB algorithm for Problem (16) when (a) $\alpha = 0.01$, (b) $\alpha = 0.05$ and (c) $\alpha = 0.1$.

both algorithms (The result is absent if no feasible solution is obtained by an algorithm). Figure

3 displays the corresponding CPU times needed for the BB algorithm to terminate. We also list some of the results in Tables 5 to 6 for more precision.

Table 5: The optimal value obtained by the BB algorithm

	$\alpha = 0.01$	$\alpha = 0.05$	$\alpha = 0.1$
v=2.5%	5.82E-03	8.61E-03	8.84E-03
v=3.5%	7.16E-03	8.84E-03	8.84E-03
v=4.5%	7.81E-03	8.84E-03	8.84E-03
v=5.5%	8.33E-03	8.84E-03	8.84E-03
v=6.5%	8.66E-03	8.84E-03	8.84E-03

Table 6: The optimal value obtained by the nonlinear optimization algorithm

	α=0.01	α=0.05	α=0.1
v=2.5%	N	8.61E-03	8.84E-03
v=3.5%	N	8.84E-03	8.84E-03
v=4.5%	7.30E-03	8.84E-03	8.84E-03
v=5.5%	8.28E-03	8.84E-03	8.84E-03
v=6.5%	8.66E-03	8.84E-03	8.84E-03

From Figure 2 and Tables 5 to 6 we can find that for various combinations of v and α, the nonlinear optimization algorithm can indeed find the optimal solutions, and at the same time, the BB algorithm verifies the global optimality. For the cases (e.g., $v = 4.5\%$ and $\alpha = 0.01$) where the nonlinear approach does not find the global optimal solutions, the BB algorithm improves the local solutions to the global optima. Note that in Table 6, "N" indicates that the algorithm does not find any feasible solutions. For the two cases, however, the BB algorithm is able to find the global optimal solutions. Figure 3 shows that it takes the BB algorithm less time to terminate when α increases. The possible reason is that the optimal values obtained by the nonlinear optimization algorithm are closer to the global optima for larger α.

Finally, in order to compare the practical impact of the specification of P_* on the optimal portfolio, we respectively fit the distribution P_* to the Gaussian mixture distribution and the conventional Gaussian distribution, and calculate the historical mean return rate and historical VaR based on the optimal solutions to Problem (15). The corresponding result is shown in Figure 4. From Figure 4, we find that when $\alpha = 0.01$, using the Gaussian mixture distribution we can obtain a better portfolio frontier. For $\alpha = 0.05, 0.1$, both distributions yield similar frontiers. This suggests that the Gaussian mixture distribution better captures the tail risks for this portfolio selection problem.

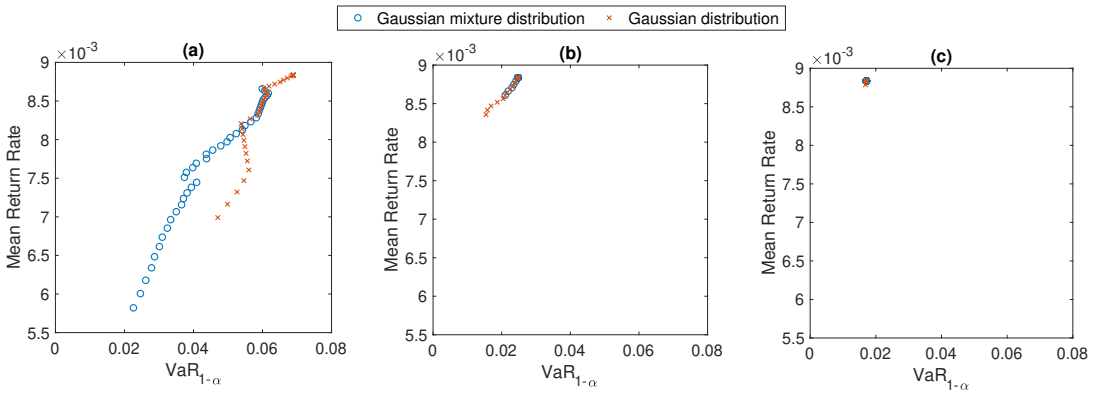

Figure 4: Risk-return pairs under different settings: Gaussian mixture distribution versus Gaussian distribution when (a) $\alpha = 0.01$, (b) $\alpha = 0.05$ and (c) $\alpha = 0.1$.

5 Conclusions

In this paper, we studied using mixture distributions to model the randomness in CCPs. We analyzed the structures and properties of CCPs with mixture distributions. Based on the results we studied several scenarios and discussed how to handle the CCPs in these scenarios. We also conducted some preliminary numerical experiments to demonstrate the implementation of our approach. We expect that the approach proposed in this paper could generalize the scope and applicability of CCPs.

Acknowledgement

This research was partially supported by Tongji University 100 Youth Program. A preliminary version of this paper was once submitted to the *Proceedings of the 2017 Winter Simulation Conference*.

References

Ahmed, S., D. J. Papageorgiou. 2013. Probabilistic set covering with correlations. *Operations Research*, **61**(2) 438-452.

Atamtürk, A., G. Berenguer, Z. M. Shen. 2012. A conic integer programming approach to stochastic joint location-inventory problems. *Operations Research*, **60**(2) 366-381.

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

Charnes, A., W. W. Cooper, G. H. Symonds. 1958. Cost horizons and certainty equivalents: An approach to stochastic programming of heating oil. *Management Science*, **4** 235-263.

Chen, W., M. Sim, J. Sun, C-P Teo. 2010. From CVaR to uncertainty set: Implications in joint chance constrained optimization. *Operations Research*, **58** 470-485.

Hanasusanto, G. A., V. Roitch, D. Kuhn, W. Wiesemann. 2017. Ambiguous joint chance constraints under mean and dispersion information. *Operations Research*, **65**(3) 751-767.

Henrion, R., A. Möller. 2012. A gradient formula for linear chance constraints under Gaussian distribution. *Mathematics of Operations Research*, **37**(3) 475-488.

Hong, L. J., Y. Yang, L. Zhang. 2011. Sequential convex approximations to joint chance constrained programs: A Monte Carlo approach. *Operations Research*, **59** 617-630.

Horst, R. 1986. A general class of branch-and-bound methods in global optimization with some new approaches for concave minimization. *Journal of Optimization Theory and Application*, **51** 271-291.

Horst, R., P. M. Pardalos, N. V. Thoai. 1995. *Introduction to Global Optimization*, Kluwer Academic Publishers, Dordrecht.

Hu, Z., L. J. Hong, L. Zhang. 2013. A smooth Monte Carlo approach to joint chance constrained program. *IIE Transactions*, **45**(7) 716-735.

Jorion, P. 2006. *Value at Risk*, Second Edition. New York: McGraw-Hill, Inc.

Law, A. M. 2013. *Simulation Modeling and Analysis*. McGraw-Hill.

Li, J. Q., A. R. Barron. 2000. Mixture density estimation. *Advances in Neural Information Processing Systems*, **12** 279-285.

Maugis-Rabusseau, C., B. Michel. 2013. Adaptive density estimation for clustering with Gaussian mixtures. *ESAIM: Probability and Statistics*, **17** 698-724.

McLachlan, G. J., D. Peel. 2000. *Finite Mixture Models*. New York: Wiley.

Miller, L. B., H. Wagner. 1965. Chance-constrained programming with joint constraints. *Operations Research*, **13** 930-945.

Nemirovski, A., A. Shapiro. 2006. Convex approximations of chance constrained programs. *SIAM Journal on Optimization*, **17** 969-996.

Pagnoncelli, B. K., S. Ahmed, A. Shapiro. 2009. Sample average approximation method for chance constrained programming: Theory and applications. *Journal of Optimization Theory and Applications*, **142** 399-416.

Parzen, E. 1962. On estimation of a probability density function and mode. *The Annals of Mathematical Statistics*, **33**(3) 1065-1076.

Pearson, K. 1894. Contributions to the mathematical theory of evolution. *Philosophical Transactions of the Royal Society of London*, A **185** 71-110.

Prékopa, A. 2003. Probabilistic programming. In *Stochastic Programming, Handbooks in OR&MS*. Vol. 10, A. Ruszczynski and A. Shapiro, eds., Elsevier.

Rosenblatt, M. 1956. Remarks on some nonparametric estimates of a density function. *The Annals*

of Mathematical Statistics, 832-837.

Sahinidis, N. V. 2017. BARON 17.8.9: Global Optimization of Mixed-Integer Nonlinear Programs, User's manual.

Sturm, J. F. 1999. Using SeDuMi 1.02, a MATLAB toolbox for optimization over symmetric cones. *Optimization Methods and Software*, **11/12** 625-653.

Sun, W., Z. Hu, L. J. Hong. 2018. Gaussian mixture model-based random search for continuous optimization via simulation. *Proceedings of the 2018 Winter Simulation Conference*, accepted.

Tawarmalani, M., N. V. Sahinidis. 2005. A polyhedral branch-and-cut approach to global optimization. *Mathematical Programming*, **103**(2) 225-249.

van Ackooij, W., R. Henrion. 2014. Gradient formulae for nonlinear probabilistic constraints with Gaussian and Gaussian-like distributions. *SIAM Journal on Optimization*, **24**(4) 1864-1889.

van Ackooij, W., R. Henrion, A. Moller, R. Zorgati. 2010. On probabilistic constraints induced by rectangular sets and multivariate normal distributions. *Mathematical Methods of Operations Research*, **71**(3) 535-549.

van Ackooij, W., R. Henrion, A. Moller, R. Zorgati. 2014. Joint chance constrained programming for hydro reservoir management. *Optimization and Engineering*, **15**(2) 509-531.

Xie, W., S. Ahmed. 2018. On quantile cuts and their closure for chance constrained optimization problems. *Mathematical Programming*, **172**(1-2) 621-646.

Wang, J., M. R. Taaffe. 2015. Multivariate mixtures of normal distributions: properties, random vector generation, fitting, and as models of market daily changes. *INFORMS Journal on Computing*, **27**(2) 193-203.

Zeevi, A. J., R. Meir. 1997. Density estimation through convex combinations of densities: approximation and estimation bounds. *Neural Networks*, **10**(1) 99-109.

Zymler, S., D. Kuhn, B. Rustem. 2013. Distributionally robust joint chance constraints with second-order moment information. *Mathematical Programming*, **137**(1-2) 167-198.