# Iterative Estimation Maximization for Stochastic Linear Programs with Conditional Value-at-Risk Constraints

Pu Huang
IBM T.J.Watson Research Center
Yorktown Heights, NY 10598

Dharmashankar Subramanian
IBM T.J.Watson Research Center
Yorktown Heights, NY 10598

**Abstract**

We present a new algorithm, Iterative Estimation Maximization (IEM), for stochastic linear programs with Conditional Value-at-Risk constraints. IEM iteratively constructs a sequence of compact-sized linear optimization problems, and solves them sequentially to find the optimal solution. The problem size IEM solves in each iteration is unaffected by the size of random samples, which makes it extreme efficient for real-world, large-scale problems. We prove that IEM converges to the true optimal solution, and give a lower bound on the number of samples required to probabilistically bound the solution error. We present computational performance on large problem instances and also conduct a case study on a financial portfolio optimization problem using an S&P 500 data set.

## 1 Introduction

Consider the following risk-constrained stochastic linear program,

$$\begin{aligned} \text{(P)} \qquad & \max_x c^T x \\ & \text{CVaR}_\beta(\xi^T x) \leq b \\ & Lx = l, x \geq 0, \end{aligned} \tag{1}$$

where $Lx = l$, $x \geq 0$, represent a set of linear inequalities, $c \in \mathbb{R}^n$ is a vector of objective function coefficients, $\xi$ is a random vector in space $\mathbb{R}^n$, $x$ is a decision vector in space $\mathbb{R}^n$, $\text{CVaR}_\beta(\cdot)$ is a risk measure called Conditional Value-at-Risk (CVaR), which maps random variables to real numbers in $\mathbb{R}$, and $b \in \mathbb{R}$ is a upper bound imposed upon the risk measure $\text{CVaR}_\beta(\cdot)$, for modeling the maximum acceptable risk that one is willing to take. In general, one could have more than one constraint with a CVaR risk measure.

Problem (P) extends the linear programming model. It allows model parameters $\xi$ to be random variables, and thus can capture uncertainty associated with the problem of interest. One example of such a problem is portfolio optimization. In a portfolio optimization problem, random vector $\xi$ represents the potential future losses of a group of assets, decision variable $x$ is the fractions of initial wealth allocated to the assets, $\text{CVaR}_\beta(\xi^T x)$ measures the risk of a given allocation $x$, $b$

bounds the maximum level of risk, $c$ typically contains the expected returns of individual assets, and budget and other constraints can be modeled as $Lx = l$, $x \geq 0$.

We concern ourselves in this paper with CVaR, a coherent risk measure proposed by Artzner et al. [1] as an alternative to Value-at-Risk (VaR), a popular risk measure widely used in financial risk management. VaR measures the *lowest* potential loss that may occur within a given small probability. Let $X$ denote a random variable with strictly-increasing and continuous probability distribution function[1], and $1 - \beta \in (0,\ 1)$ denote a small probability that a large loss may occur, then VaR can be defined as

$$\mathrm{VaR}_\beta(X) = \min\{x \in \mathbb{R} : \Pr\{X \leq x\} \geq \beta\},$$

and CVaR can be represented in terms of VaR as

$$\mathrm{CVaR}_\beta(X) = \mathbb{E}[X | X \geq \mathrm{VaR}_\beta(X)].$$

In essence, CVaR measures the average extreme loss under the condition that the actual loss is greater than VaR.

If we replace CVaR in constraint (1) by VaR, we get a chance-constrained stochastic linear program, which has been studied extensively (see, e.g. [6] and references therein). One major difficulty of applying such a chance constraint to bound risk is that in general it leads to a non-convex problem and hence is difficult to solve.

Unlike its chance-constrained variant, Problem (P) is convex, due of the nice properties that CVaR enjoys. Following Artzer et al. [1], CVaR is a coherent risk measure and thus satisfies the following four properties that any coherent risk measure $\rho(\cdot)$ does:

- Translation invariance. For any random variable $Y$ and any real number $k$, $\rho(Y + k) = \rho(Y) + k$.
- Sub-additivity. For any random variables $X$ and $Y$, $\rho(X + Y) \leq \rho(X) + \rho(Y)$.
- Positive homogeneity. For any random variable $Y$ and any positive real number $\lambda \geq 0$, $\rho(\lambda Y) = \lambda \rho(Y)$.
- Monotonicity. For any random variables $X$ and $Y$, if $X \leq Y$ almost surely, then $\rho(X) \leq \rho(Y)$.

Sub-additivity and positive homogeneity implies convexity and therefore that the feasible region defined by constraint (1) is convex. This means that, potentially, large instances of Problem (P) can be solved efficiently. However, although it is a convex problem, finding the exact solution to Problem (P) is generally impossible. This is because even for a given $x$, computing $\mathrm{CVaR}_\beta(\xi^T x)$, the l.h.s. of constraint (1), requires evaluating an $n$-dimensional integral, which, in general, can only be approximated when $n$ is large.

[1]The definition of CVaR for general distribution functions is more technically involving (see [8] for details). However, it poses no extract difficulties to our algorithm as what we in essence need is merely a CVaR estimator to embed into our algorithm. We make this assumption purely for presentation simplicity.

In this paper, we develop an efficient algorithm, Iterative Estimation Maximization (IEM), for Problem (P). IEM solves a sample approximation problem of the original Problem (P), where random vector $\xi$ is replaced by a set of random samples of $\xi$. IEM iteratively constructs a sequence of compact-sized linear programs and solves them sequentially. The solution obtained in the preceding iteration is used to construct the linear program for the next iteration in such a way that it guarantees the convergence of the algorithm. The size of the linear program that IEM solves in each iteration is comparable to the original problem, and is unaffected by the size of random samples, which makes it extremely efficient for large problem instances. Indeed, our numerical experiments showed that IEM solves large problem instances an order of magnitude faster than another wildly used algorithm proposed by Rockafellar and Uryasev [7].

We analyze the performance of IEM from a column generation point of view and show that IEM indeed solves a linear program with exponential number of constraints using column generation. From this perspective, IEM constructs the column generation sub-problems in such a way that they can be solved by sorting $N$ values, where is $N$ is number of samples of random vector $\xi$, and thus each column generation sub-problem can be solved in $O(N \log N)$ time. This explains the good practical performance of IEM as column generation has been well known for its efficiency in solving large linear programs. Considering that we are solving a stochastic linear program with $N$ samples, the extra overhead of $O(N \log N)$ seems to be unavoidable.

Although we present IEM in this paper for Problem (P) with a single CVaR constraint (1), it is straightforward to extend IEM to handle multiple constraints. IEM can also be used to solve problems with a CVaR objective, as such an objective can be easily transformed into a CVaR constraint by introducing an auxiliary variable.

Rockafellar and Uryasev [7] proposed a sample approximation formulation for Problem (P). They approximated the CVaR constraint (1) using the following constraint

$$t + \frac{1}{N(1-\beta)} \sum_{i=1}^{N} (\xi_i^T x - t)^+ \leq b,$$

where $\xi_1, \xi_2, ..., \xi_N$ are $N$ samples of random vector $\xi$, and $t$ is an auxiliary variable. Introducing variables, $z_i$, $i = 1, 2, ..., N$, they linearized the above constraint as the following set of constraints

$$\begin{aligned} t + \frac{1}{N(1-\beta)} \sum_{i=1}^{N} z_i &\leq b \\ z_i \geq \xi_i^T x - t, \quad & i = 1, 2, ..., N \\ z_i \geq 0, \quad & i = 1, 2, ..., N. \end{aligned}$$

This procedure transforms Problem (P) to a large linear program and solves it accordingly. One major shortcoming of this approach is that both the number of auxiliary variables, $z_i$, and the number of newly introduced constraints are proportional to the number of samples, which makes this approach computationally challenging even for modest size original problems, when the sample size is large. Our IEM algorithm does not suffer from this shortcoming. More recently, Künzi-Bay and Mayer [3] retained the same sample approximation formulation and recasted the CVaR

optimization problem as a two-stage stochastic program. They specialized the L-shaped method to develop a new algorithm for minimizing CVaR. Their algorithm also led to a significant improvement in computational performance, over the formulation of Rockafellar and Uryasev. However their algorithm can only handle CVaR objective. For problems with multiple CVaR constraints, it is not immediately clear how they can lend themselves to a two-stage interpretation.

The rest of the paper is organized as follows. Section 2 presents the IEM algorithm for Problem (P) and proves its correctness. In Section 3 we give a lower bound on sample size $N$ so that the sample approximation problem IEM solves closely approximates the original problem with high probability. Section 4 shows the performance of IEM on randomly generated problem instances. Section 5 describes numerical results on a practical portfolio optimization problem over financial securities. Section 6 concludes the paper.

## 2 Iterative Estimation Maximization (IEM) Algorithm

We first describe the IEM algorithm.

---

**Iterative Estimation Maximization:**

1. Generate $N$ samples $\{\xi_1, \xi_2, ..., \xi_N\}$ of random vector $\xi$.

2. Set iteration index $t = 0$. Let $LP^0$ denote the initial linear program, $\{\max_x c^T x : \ Lx = l, x \geq 0\}$, in iteration $t = 0$. Solve $LP^0$ to get an initial value $x^0$. If $LP^0$ is unbounded, assign $x^0$ a random value. If $LP^0$ is infeasible, terminate and declare infeasibility.

3. Let $LP^t$ and $x^t$ denote the linear program and the optimal solution in the current iteration $t$. Perform the following steps:

   (a) Compute $L_i = \xi_i^T x^t$ for each sample $\xi_i, \ i = 1, 2, ..., N$.

   (b) Sort $L_i$, $i = 1, 2, ..., N$, in ascending order. Let $L_{(i)}$ denote the $i$-th smallest value in the list of sorted $L_i$ values, and $\xi_{(i)}$ denote the corresponding sample used to compute $L_{(i)}$, i.e.,
   $$\xi_{(1)}^T x^t \leq \xi_{(2)}^T x^t \leq ... \leq \xi_{(N)}^T x^t.$$

   (c) Compute
   $$H(x^t) = (N - K + 1)^{-1} \sum_{i=K}^{N} \xi_{(i)}^T x^t, \tag{2}$$
   where $K = \lceil N\beta \rceil$ is the smallest integer that is greater than $N\beta$. If $H(x^t) \leq b$, terminate and output $x^* = x^t$ as the optimal solution. Otherwise, add the following constraint to the current linear program $LP^t$,
   $$(N - K + 1)^{-1} \sum_{i=K}^{N} \xi_{(i)}^T x \leq b. \tag{3}$$

4. We now have a new linear program $LP^{t+1}$ after adding a new constraint (3) to $LP^t$. Solve $LP^{t+1}$ and get a new optimal solution $x^{t+1}$. If $LP^{t+1}$ is unbounded, assign $x^{t+1}$ a random value. If $LP^{t+1}$ is infeasible, terminate and declare infeasibility.

5. Set $t = t + 1$ and go back to step 3.

---

For a given solution $x^t$, IEM estimates the value of $\text{CVaR}_\beta(\xi^T x^t)$ as $H(x^t)$, which is computed based on samples $\{\xi_1^T x^t, \xi_2^T x^t, ..., \xi_N^T x^t\}$. In step 3c, those $\xi_i$'s that have contributed to the computation of $H(x^t)$, i.e., the top $N - K + 1$ samples, are used to create constraint (3), which are then added to the next iteration optimization problem, $LP^{t+1}$. IEM compares $H(x^t)$ and $b$ in each iteration, and if $H(x^t) \leq b$, the algorithm terminates; otherwise, it continues to the next iteration. Intuitively, IEM seeks to iteratively approximate the CVaR constraint (1) in the original Problem (P) using linear constraints as shown in (3). It does so by using the best solution $x^t$ obtained so far to estimate the coefficients of the linear constraint (3). When IEM terminates, $\text{CVaR}_\beta(\xi^T x^*)$, the CVaR value of $\xi^T x^*$ corresponding to the optimal solution $x^*$, is approximated by $H(x^*)$, which satisfies the upper bound $b$.

We now show that IEM indeed solves a sample approximation problem of the original Problem (P).

Let $\mathscr{C}$ denote the collection of all subsets of the set, $\{\xi_1, \xi_2, ..., \xi_N\}$, with size, $N - K + 1$, and $I_A(\cdot)$ denote the indicator function of a subset $A \in \mathscr{C}$, i.e.,

$$I_A(a) = \begin{cases} 1 & a \in A \\ 0 & a \notin A \end{cases}.$$

Corresponding to each subset $A \in \mathscr{C}$, we can construct a linear constraint that take the following form,

$$(N - K + 1)^{-1} \sum_{i=1}^{N} \xi_i^T x I_A(\xi_i) \leq b.$$

Constructing a linear constraint like the above one for each $A \in \mathscr{C}$, leads to an exponentially large set of constraints. Theorem 1 asserts that the original Problem (P) can be approximated by replacing the CVaR constraint (1) with such an exponential set of constraints, leading to a sample approximation problem. Theorem 2 shows that IEM solves the sample approximation problem and converges after a finite number of iterations.

**Theorem 1.** *Given $N$ samples $\{\xi_1, \xi_2, ..., \xi_N\}$ of random vector $\xi$, the following linear program (SA) is a sample approximation problem of the original Problem (P).*

$$\begin{aligned} (SA) \qquad & \max_x c^T x \\ & (N - K + 1)^{-1} \sum_{i=1}^{N} \xi_i^T x I_A(\xi_i) \leq b, \quad \forall A \in \mathscr{C} \qquad (4) \\ & Lx = l, x \geq 0. \end{aligned}$$

*Proof.* For $\forall x \in \mathbb{R}^n$, $\xi_i^T x$, $i = 1, 2, ..., N$, can be viewed as a set of samples of random variable $\xi^T x$. Sort these samples in an ascending order as $\xi_{(1)}^T x \leq \xi_{(2)}^T x \leq ... \leq \xi_{(N)}^T x$. Given these sorted samples, an aymptotically unbiased estimator for $\text{CVaR}(\xi^T x)$ is given as (see, e.g. [4]),

$$(N - K + 1)^{-1} \sum_{i=K}^{N} \xi_{(i)}^T x, \tag{5}$$

where $K = \lceil N\beta \rceil$ is the smallest integer that is greater than $N\beta$.

Define the dominant set $B(x)$ induced by a given $x$ as

$$B(x) = \{A \in \mathscr{C} : \ \xi_i^T x \geq \xi_j^T x, \quad \forall \xi_i \in A, \ \forall \xi_j \notin A\}, \tag{6}$$

i.e., $B(x) \in \mathscr{C}$ contains $N - K + 1$ samples from $\xi_i$, $i = 1, 2, ..., N$, such that for any $\xi_i \in B(x)$ and any $\xi_j \notin B(x)$, $\xi_i^T x \geq \xi_j^T x$. In essence, $B(x)$ contains the top $N - K - 1$ samples with the largest $\xi_i^T x$ value.

Use $B(x)$ to represent the CVaR estimator as shown in (5), plug it back into inequality (1), we have the following approximation to constraint (1)

$$(N - K + 1)^{-1} \sum_{i=1}^{N} \xi_i^T x I_{B(x)}(\xi_i) \leq b. \tag{7}$$

Now consider the rest of constraints in constraint set (4),

$$(N - K + 1)^{-1} \sum_{i=1}^{N} \xi_i^T x I_C(\xi_i) \leq b, \quad \forall C \in \mathscr{C} - B(x). \tag{8}$$

We claim that constraint (7) implies all constraints in (8). This is because

$$\sum_{i=1}^{N} \xi_i^T x I_C(\xi_i) \leq \sum_{i=1}^{N} \xi_i^T x I_{B(x)}(\xi_i), \quad \forall C \in \mathscr{C} - B(x),$$

as replacing any $\xi_i \in B(x)$ by any $\xi_j \notin B$ always results in a smaller, or equal, $\xi_i^T x$ value. Constraint (7) is thus equivalent to the constraint set (4), and thus Problem (SA) is a sampling approximation to the original Problem (P). □

**Remarks.** Note that constraint (3) belongs to constraint set (4). In iteration $t$, it is the constraint in set (4) that is indexed by $B(x^t)$, the dominant set induced by $x^t$.

**Theorem 2.** *IEM solves Problem (SA).*

*Proof.* Consider the following dual of Problem (SA)

$$\begin{aligned} (D) \qquad & \min_{\lambda_A, y} \Big( \sum_{A \in \mathscr{C}} b\lambda_A \Big) + l^T y \\ & (N - K + 1)^{-1} \Big( \sum_{A \in \mathscr{C}} \lambda_A \sum_{i=1}^{N} \xi_i I_A(\xi_i) \Big) + L^T y \geq c \\ & \lambda_A \geq 0, \quad \forall A \in \mathscr{C}, \end{aligned} \tag{9}$$

where $\lambda_A$ is the dual variable of the constraint indexed by $A \in \mathscr{C}$ in constraint set (4), and $y$ is the vector of free dual variables corresponding to the equality set $Lx = l$. Notice that Problem (D) has a combinatorially large number of decision variables $\lambda_A$'s (more precisely, $C^N_{N-K+1}$), and a natural way to solve it is a column generation approach. In such an approach, Problem (D) is decomposed into a master problem and a slave problem, and the solution procedure iterates between these two sub-problems. In each iteration $t$, the solution of the master problem is fed into the slave problem, which in turn generates a column (data associated with a $\lambda_A$) by minimizing a "reduced-cost" of the master problem, and feeds the generated column back to the master problem. Consider the following column generation decomposition scheme.

- The master problem takes exactly the same form as Problem (D) except that it only contains a small subset of decision variables $\lambda_A$'s. More precisely, replacing $\mathscr{C}$ in Problem (D) by $\mathscr{D}^t$, where $\mathscr{D}^t \subset \mathscr{C}$ and $|\mathscr{D}^t| \ll |\mathscr{C}|$ is a small subset of $\mathscr{C}$, we get the master problem in iteration $t$. The initial subset $\mathscr{D}^0$ is empty.

- Let $x^t$ denote the *primal* solution to the master problem in iteration $t$. ($x^t$ are the primal variables corresponding to constraint (9) in the master problem.) The slave problem that seeks to identify the non-basic column that minimizes the "reduced-cost" of the master problem, takes the following form,

$$\{\min_A \sum_{i=1}^{N} b - \xi_i^T x^t I_A(\xi_i),\ \forall A \in \mathscr{C}\}. \tag{10}$$

It is clear that for any given $x^t$, the dominant set $B(x^t)$ induced by $x^t$ solves the slave problem (10). Finding $B(x^t)$ is easy, as we only need to compute $\xi_i^T x^t$ for each $\xi_i,\ i = 1, 2, ..., N$, sort them, and pick up the top $\xi_i$'s with the largest $\xi_i^T x^t$ values. This is exactly what IEM does in steps 3a and 3b.

After obtain $B(x^t)$, the column generation procedure adds $B(x^t)$ to $\mathscr{D}^t$ (i.e., adds a new decision variable $\lambda_{B(x^t)}$ to the master problem) and continues to the next iteration. Adding a new variable in the dual is equivalent to adding a new constraint in the primal. This is what IEM does in step 3c.

Also in step 3c, IEM compares $b$ and $H(x^t)$ to decide whether to terminate, or continue to the next iteration. This can be understood in terms of using the simplex algorithm to solve the master problem. Note that $b - H(x^t)$ is the "reduced cost" of decision variable $\lambda_{B(x^t)}$ in the master problem. If $b - H(x^t) \geq 0$, then the reduced costs corresponding to any other decision variable $\lambda_A,\ A \in \mathscr{C}$, is non-negative as well. This is because $B(x^t)$ solves the slave problem (10) and thus $b - H(x^t)$ is the smallest possible "reduced cost". In such a case, we have already optimally solved the master problem and correspondingly the dual Problem (D), and therefore $x^t$ is the optimal solution to the primal Problem (SA). □

**Remarks.** We would like to bring attention to a few subtle issues related to the IEM algorithm.

- Removal of non-binding constraints. Consider using the simplex algorithm to solve the master problem. After identifying $B(x^t)$, and adding $\lambda_{B(x^t)}$ into the master problem, the simplex method will select $\lambda_{B(x^t)}$ to enter the basis. All of the rest dual variables $\lambda_A$'s in the master problem that are not in the basis will continue to be kept as zeros. Those dual variables that are continued to be kept as zeros correspond to non-binding constraints in the primal, and thus can be safely removed without affecting the solution of the next iteration. Therefore, we can remove all non-binding constrains in step 4 of the IEM algorithm before solving the next iteration problem $LP^{t+1}$. However, doing so may cause IEM to cycle indefinitely if the removed constraints are added back later and then removed again. Methods that can prevent the simplex algorithm from cycling are well-studied and well-known. One such a method is lexicographic ordering. A lexicographic order on the dual variables $\lambda_A$'s in the master problem defines a lexicographic order on the constraints (4) in the primal Problem (SA). If we follow such an order to break a tie when adding a new constraint (3) in IEM step 3c, we can prevent IEM from cycling. Practically speaking, removing non-binding constraints may not save much computational time, as IEM only adds one constraint in each iteration, and it typical finds the optimal solution in a small number of iterations.

- Unbounded case. If $LP^0$ is unbounded, we can choose any feasible $x^0$ to start IEM. In the sequence of IEM iterations, it may happen that problem $LP^{t+1}$ is unbounded as well. In such a case, any feasible solution $x^{t+1}$ can be chosen to continue the algorithm. However, we need to make sure that $x^{t+1}$ will not keep re-introducing existing constraints to problem $LP^{t+1}$. One simple way to guarantee that is by randomly selecting a feasible $x^{t+1}$.

- Mixed-integer stochastic program. IEM also solves CVaR-constrained mixed-integer stochastic programs. When applying IEM to such a problem, we need to solve a Mixed-Integer Program (MIP), instead of a linear program, in each iteration $t$. As finding the optimal solution to a MIP itself is a hard problem in general, it may take a long time for IEM to converge as it iteratively solves MIPs. An alternative more efficient approach is as follows: First, ignore all the integerality constraints, and apply IEM to solve the LP-relaxation of the sampling approximation problem. Second, upon successful termination of IEM on the LP-relaxation, keep all the constraints accumulated so far, turn on the integrality constraints, and start IEM once again to iteratively solve MIPs until convergence. In this approach, we first accumulate constraints to shrink the feasible region to save some computational time, and then use IEM to solve the original problem with integerality constraints.

# 3 Asymptotic Bounds

The following theorem asserts that the CVaR estimator $H(\cdot)$ asymptotically converges to the "true" value as $N$ increases.

**Theorem 3.** *(Nagaraja) Assume $\xi^T x$ has finite mean and variance for any $x$, then $\sqrt{N(1-\beta)}[H(x) - \text{CVaR}_\beta(\xi^T x)]$ converges in distribution to a normal distribution with mean $\mu = 0$ and variance*

$\sigma_x^2 = \sigma_\beta^2 + \beta[\mathrm{CVaR}_\beta(\xi^T x) - \mathrm{VaR}_\beta(\xi^T x)]^2$, *where* $\sigma_\beta^2$ *is the variance of the distribution of* $\xi^T x$ *truncated at* $\mathrm{VaR}_\beta(\xi^T x)$.

*Proof.* CVaR estimation is a special case of the "trimmed mean" estimation problem. Proof for the above asymptotic result can be found in [5]. For a more recent treatment using influence functions, see [4]. □

Let $R^\epsilon$ denote the feasible region of constraint set (4) with a small perturbation $\epsilon > 0$ on the r.h.s. $b$, i.e.,

$$R^\epsilon = \{x \mid (N-K+1)^{-1} \sum_{i=1}^{N} \xi_i^T x I_A(\xi_i) \leq b + \epsilon, \quad \forall A \in \mathscr{C}\},$$

and $R$ denote the feasible region of the original CVaR constraint (1), i.e.,

$$R = \{x \mid \mathrm{CVaR}_\beta(\xi^T x) \leq b\},$$

we would like to give a bound on $N$ such that

$$\Pr\{R^{-\epsilon} \subseteq R \subseteq R^\epsilon\} \geq 1 - 2\delta, \tag{11}$$

where $\delta$ is a small probability.

If for any $x \in R^{-\epsilon}$, $x$ also satisfies inequality $-\epsilon \leq H(x) - \mathrm{CVaR}_\beta(\xi^T x)$, we know that $R^{-\epsilon} \subseteq R$. Similarly, $x \in R$ and $x$ satisfying $H(x) - \mathrm{CVaR}_\beta(\xi^T x) \leq \epsilon$ imply $R \subseteq R^\epsilon$. Therefore, it is sufficient for inequality (11) to hold if the following is true,

$$\Pr\{-\epsilon \leq H(x) - \mathrm{CVaR}_\beta(\xi^T x) \leq \epsilon\} \geq 1 - 2\delta, \quad \forall x \in R. \tag{12}$$

The following point-wise bound $N_x$ (depending on $x$) on sample size $N$ is an immediate conclusion of Theorem 3,

$$N_x \geq \frac{\sigma_x^2 \Phi^{-1}(1-\delta)}{\epsilon^2(1-\beta)}, \quad \forall x \in R,$$

where $\Phi^{-1}(\cdot)$ is the inverse of the CDF of the standard the normal distribution. For practical problems, it is often not hard to get a uniform bound. For example, if $\xi$ has a bounded support with diameter $D_\xi$, and $x$ is bounded by $D_x$, a (very conservative) uniform lower bound is as follows,

$$N > \frac{(1+\beta) D_\xi^2 D_x^2 \Phi^{-1}(1-\delta)}{\epsilon^2(1-\beta)}.$$

The above result is obtained by bounding $\sigma_x^2$ using its maximum possible value $(1+\beta)D_\xi^2 D_x^2$.

Asymptotic bounds in general should only be used as a guideline to decide how many samples to generate in order to achieve certain solution accuracy. In practical, they may not be tight enough to give an accurate estimation $N$. Experimenting to find the suitable $N$ value should always be carried out for a particular problem.

## 4 Computational Performance

In order to characterize the computational performance of IEM, we conducted a comparison study on a set of randomly generated problem instances. In this section, we describe the random problem instances and report the observed computational performance of IEM, in comparison with the linear programming reformulation proposed by Rockafellar and Uryasev [7].

The random problem instances takes the following form,

$$\begin{aligned} &\max_x c^T x \\ &\text{CVaR}_\beta(\sum_{i\in I} a_{ji}x_i) \le b_j, \quad j = 1, 2, ...M, \\ &0 \le x_i \le 1, \quad i = 1, 2, ..., V, \end{aligned} \tag{13}$$

where $x$ is the decision variable vector, indexed over set, $I = \{1, 2, ..., V\}$, with $V$ being fixed to 30. The constraint set (13) contains a group of CVaR constraints, indexed over set $J = \{1, 2, ..., M\}$, all with $\beta = 0.9$. The stochastic parameters, $a_{ji}$, were generated as follows,

$$a_{ji} = \max(0.1, \text{Normal}(\text{Uniform}(1,10), \text{Uniform}(5,10))).$$

The objective function coefficient vector, $c$ was fixed, taking values in the range [1,10]. The constraint r.h.s. coefficients, $b_j$ were fixed to 1 for each constraint $j \in J$.

We varied $M$ from 2 to 200. For each $M$ value, we generate 10 different problem instances, and compare IEM with Rockafellar and Uryasev's method over all these 10 instances. For each instance, we generate 1000 samples for parameters $a_{ji}$.

For fair comparison, we implemented both IEM and Rockafellar and Uryasev's method using AMPL. For IEM, the sorting procedure in Step 3 is implemented in C and hooked into AMPL. We used ILOG CPLEX 10.0 as the underlying linear programming solver for both algorithms. The computations were performed on a server with an Intel Xeon CPU, at 2.70 GHz, and 3.5 GB RAM.

The table below shows the average and standard deviation of computational times of both algorithm across the 10 random instances, for each value of $M$. Rockafellar and Uryasev's method took 22175.8 seconds to finish a single instance of $M = 200$. Due to the long computational time, we only ran a single instance of $M = 200$ for their algorithm.

The results in Table 1 are visualized in Figure 1. It clearly shows that the performance of IEM is significantly superior when the number of constraints is large. This is because, as mentioned earlier, Rockafellar and Uryasev's method introduces a large number of additional variables and constraints that are proportional to the number of samples for each original CVaR constraint in Problem (P). When the number of CVaR constraints increases, the additional complexity significantly impacts its performance. On the other hand, IEM does not suffer from such an increase in complexity, because in each iteration of IEM, we solve a linear programming problem that is almost the same size as the original Problem (P). This aspect of IEM is very appealing from both a computational time, as well as computer memory size point of view. It is also very appealing in terms of the sample average approximation quality, because the algorithmic complexity of IEM is relatively insensitive

| Instance | M | Rockafellar and Uryasev's method | | IEM | |
|---|---|---|---|---|---|
| | | Mean | Std. Deviation | Mean | Std. Deviation |
| 1 | 2 | 1.33 | 0.07 | 53.73 | 9.76 |
| 2 | 10 | 36.41 | 4.30 | 116.71 | 21.07 |
| 3 | 50 | 773.82 | 101.94 | 350.71 | 77.79 |
| 4 | 100 | 3134.78 | 307.83 | 694.08 | 121.28 |
| 5 | 150 | 8746.12 | 1990.80 | 868.77 | 184.43 |
| 6 | 200 | 22175.80 | - | 1309.33 | 260.54 |

Table 1: Comparing performance of IEM and Rockafellar and Uryasev's method.

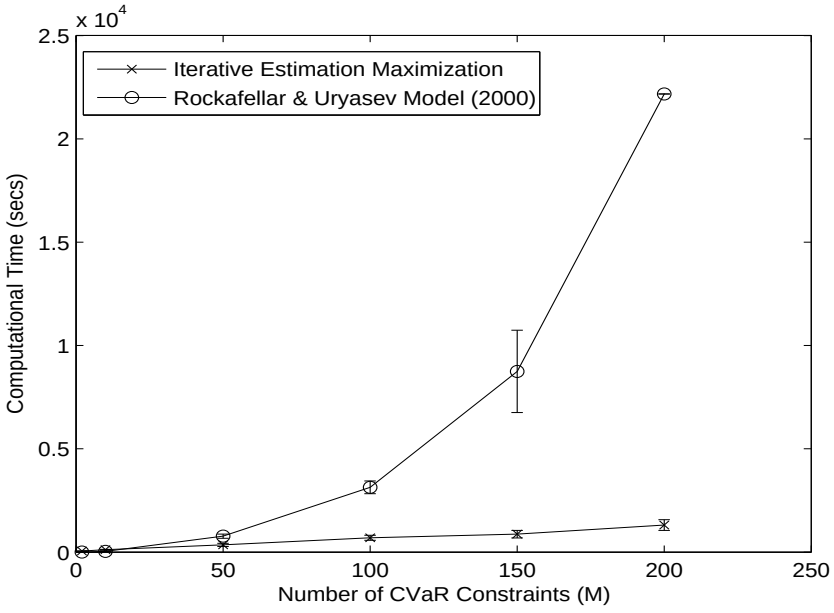


Figure 1: Comparison of performance of IEM and Rockafellar and Uryasev's method.

to the number of samples and thus we can use a large size of samples to better approximate the original problem.

# 5 A Financial Portfolio Selection Application

In this section, we study a financial portfolio application over a set of 493 stocks from the S&P 500 list (as of December 26th, 2008), excluding SNI, DPS, PM, TEL, TDC, DFS and COV, due to insufficient historical data. Our study is similar to that of Krokhmal et al. [2] except that we impose additional CVaR constraints on individual industrial sectors, as well as on overall portfolio. We consider a data set consisting of daily closing prices for these 493 stocks, over a one-year period spanning from May 23rd 2007 to May 22nd 2008, obtained from *http://finance.google.com* with adjustment for stock splitting. These stocks span a total of 10 industrial sectors as shown in Table 2. We first describe our method of generating radom samples required by IEM, then present our model formulations and computational results.

| Sector | Name | Number of Stocks |
|---|---|---|
| 1 | Consumer Discretionary | 80 |
| 2 | Consumer Staples | 41 |
| 3 | Energy | 39 |
| 4 | Financials | 84 |
| 5 | Health Care | 54 |
| 6 | Industrials | 58 |
| 7 | Information Technology | 74 |
| 8 | Materials | 28 |
| 9 | Telecommunications Services | 9 |
| 10 | Utilities | 33 |

Table 2: S&P 500 Sectors.

## 5.1 Sample Generation

Let $I = \{1, ..., V\}$ denote the set of candidate stocks, and let $J = \{1, ..., M\}$ denote the set of sectors. Set $I$ partitions across set $J$, and let these partitions be denoted by $I_j$, i.e., a subset of $I$ that contains all the tickers belonging to sector $j$. Note that $V = 493$ and $M = 10$ in our analysis. Further, let $x_i$, $i \in I$, denote the fraction of the total capital (normalized to unity) that is allocated to each instrument $i$, and $r_i$ denote the corresponding end-of-horizon return ratio. The distribution of random vector $r = [r_1, r_2, ..., r_V]$ is estimated based on historical data, and the samples required by IEM are drawn from the estimated distribution. We assume that the logarithm of $r$, i.e. $[\ln(r_1), ..., \ln(r_V)]$, follows a multi-variate normal distribution, and estimate the corresponding mean and covariance matrix based on historical data.

Let $p_{t,i}$ denote the closing price of ticker $i$ on day $t$, we construct a new dataset

$$d_{t,i} \quad = \quad \ln(p_{t+\Delta,i}/p_{t,i})$$

for each ticker $i$ and each time point $t$, where $\Delta$ is a moving-window that slides over the data on the time dimension $t$. Note that $d_{t,i}$ is the logarithmic return ratio of ticker $i$ starting from day $t$ over $\Delta$ trading days. We use $d_{t,i}$ to estimate the mean and covariance matrix, generate $N$ samples of $[\ln(r_1), ..., \ln(r_V)]$, exponentiate these samples to get samples of return vector $r$. In our analysis, $\Delta = 30$ days, and $N = 1000$.

### 5.1.1 Model Formulations and Results

We first consider the following model.

$$\begin{aligned}
\text{(M1)} \quad & \min z \\
& \text{CVaR}_\beta(-r^T x) \le z && (14)\\
& \sum_{i \in I} x_i = 1.0 && (15)\\
& l\delta_i \le x_i \le u\delta_i, \quad \forall i \in I && (16)\\
& n_l \le \sum_{i \in I} \delta_i \le n_u && (17)\\
& s_l \le \sum_{j \in J} \kappa_j \le s_u && (18)\\
& C_j \kappa_j \le \sum_{i \in I_j} \delta_i \le D_j \kappa_j, \quad \forall j \in J && (19)\\
& L_j \kappa_j \le \sum_{i \in I_j} x_i \le U_j \kappa_j, \quad \forall j \in J && (20)\\
& \delta_i \in \{0,1\}, \ \forall i \in I; \quad \kappa_j \in \{0,1\}, \ \forall j \in J. && (21)
\end{aligned}$$

The auxiliary variable $z$ in Model (M1) sets the upper bound of overall risk of portfolio loss, measured by $\text{CVaR}_\beta(-r^T x)$, and we want to minimize that risk. Budget is scaled to unity as shown in constraint (15). $\delta_i$ is a 0-1 variable indicating the selection of ticker $i$. Constraint (16) models the minimum ($l$) and maximum ($u$) amounts of allocation that are allowed for any ticker $i$ if it is selected. Constraint (17) sets the lower ($n_l$) and upper ($n_u$) bounds on the total number of tickers. $\kappa_j$ indicates if a sector $j$ is selected into the portfolio. Constraint (18) sets the minimum ($s_l$) and maximum ($s_u$) number of sectors that are allowed in any feasible portfolio. Constraints (19) sets the minimum ($C_j$) and maximum ($D_j$) number of tickers that may be selected from each sector $j$, and constraint (20) sets the minimum ($L_j$) and maximum ($U_j$) investment that is allowed in each sector $j$.

Constraints (16)-(21) are used to model user-specified preferences with respect to the desired diversification of the portfolio. In this exercise, we set $\beta = 95\%$, $l = 0.005$, $u = 1.0$, $n_l = 0$, $n_u = 493$, $s_l = 0$, $s_u = 10$, $C_j = 0$, $D_j = |I_j|$, $L_j = 0$, and $U_j = 1$. The output of Model

(M1) is shown in Table 3. The second column of the table shows the number of tickers selected from each sector. The third column is the budget allocation $x_i$ in percentage. The optimal value $z^* = -1.0081$, which should be interpreted as follows: the average portfolio return below the 5% ($1 - \beta = 5\%$) quantile is 100.81%, i.e. the *average worst-case* return ratio is 100.81%.

| Sector | Number of Tickers | Allocation ($x_i$) |
|---|---|---|
| 1 | 1 | 4.76% |
| 2 | 0 | 0.00% |
| 3 | 2 | 13.88% |
| 4 | 1 | 18.78% |
| 5 | 2 | 20.77% |
| 6 | 2 | 15.47% |
| 7 | 0 | 0.00% |
| 8 | 2 | 26.34% |
| 9 | 0 | 0.00% |
| 10 | 0 | 0.00% |
| Whole Portfolio | 10 | 100.00% |

Table 3: Minimize CVaR risk of overall portfolio loss.

The same idea of minimizing the risk of portfolio loss can be applied to individual sectors as well. This leads to the following Model (M2) for each sector $j \in J$.

$$
\begin{aligned}
\text{(M2)} \quad & \min z_j \\
& \text{CVaR}_\beta(-\sum_{i \in I_j} r_i x_i) \le z_j \\
& \sum_{i \in I_j} x_i = 1.0 \\
& l\delta_i \le x_i \le u\delta_i, \quad \forall i \in I_j \\
& n_l \le \sum_{i \in I_j} \delta_i \le n_u \\
& \delta_i \in \{0, 1\}, \quad \forall i \in I_j.
\end{aligned}
$$

Note that Model (M2) takes stocks in a given sector $j$ as the available candidates to select from. We set $\beta = 95\%$, $l = 0.005$, $u = 1.0$, $n_l = 0$, and $n_u = |I_j|$, run Model (M2) for each sector. Table 4 reports the value of $-z_j^*$ (i.e., average worst-case return below 5% quantile) for each sector $j$.

The values of $z_j^*$ are used to set up the parameters in Model (M3), which minimizes the risk of overall portfolio loss as in Model (M1), and at the same time bounds the risk of loss in each sector.

$$
\begin{aligned}
\text{(M3)} \quad & \min z \\
& \text{Constraints (14) - (21)} \\
& \text{CVaR}_\beta(\sum_{i \in I_j} -r_i x_i) \le \omega_j \sum_{i \in I_j} x_i, \quad \forall j \in J \qquad (22)
\end{aligned}
$$

| Sector | $-z_j^*$ |
|---|---|
| 1 | 95.85% |
| 2 | 96.39% |
| 3 | 92.78% |
| 4 | 96.11% |
| 5 | 97.68% |
| 6 | 97.75% |
| 7 | 94.54% |
| 8 | 97.45% |
| 9 | 88.62% |
| 10 | 94.44% |

Table 4: Average portfolio return below 5% quantile for each individual sector.

The additional constraints (22) in Model (M3) insist that the CVaR of portfolio loss in each sector $j$, should be no worse than a user-defined percentage, $\omega_j$, relative to the budget allocated to that sector. We set $\beta = 95\%$, $l = 0.005$, $u = 1.0$, $n_l = 0$, $n_u = 493$, $s_l = 0$, $s_u = 10$, $C_j = 0$, $D_j = |I_j|$, $L_j = 0$, $U_j = 1$, and $\omega_j = z_j^*$, run Model (M3). Table 5 reports the results. The optimal value of Model (M3) is $-98.38\%$.

| Sector | Number of Tickers | Allocation ($x_i$) |
|---|---|---|
| 1 | 0 | 0.00% |
| 2 | 0 | 0.00% |
| 3 | 0 | 0.00% |
| 4 | 6 | 24.06% |
| 5 | 0 | 0.00% |
| 6 | 8 | 42.04% |
| 7 | 0 | 0.00% |
| 8 | 5 | 33.90% |
| 9 | 0 | 0.00% |
| 10 | 0 | 0.00% |
| Whole Portfolio | 19 | 100.00% |

Table 5: Minimize CVaR risk of overall portfolio loss with additional bounds on risk exposure to individual sectors.

So far we have illustrated various types of portfolio analysis that are enabled by IEM. Lastly, we generate an efficient frontier that captures the trade-off between the expected average return and the CVaR risk of a portfolio. We use Model (M3) to compute the upper bound on the CVaR risk of portfolio loss (i.e., $\text{CVaR}_\beta(-r^T x)$). We set the parameters to $\beta = 95\%$, $l = 0.005$, $u = 1.0$, $n_l = 10$, $n_u = 493$, $s_l = 5$, $s_u = 10$, $C_j = 0$, $D_j = |I_j|$, $L_j = 0$, $U_j = 1$, and $\omega_j = 0.95 z_j^*$. The

optimal $z$ value is $-100.52\%$, i.e., the worst-case return ratio is 100.52%. The corresponding mean return ratio is 105.49%. This pair of worst-case/mean return ratios give one extreme point on the efficient frontier. To get anther extreme point, we restore to the following Model (M4).

$$\begin{aligned} \text{(M4)} \qquad & \max \mathbb{E}(r^T x) \\ & \text{Constraints (15) - (22)} \end{aligned}$$

Model (M4) maximizes the expected mean return with constraints on risk exposures in individual sectors. Note that it does not bound the overall portfolio CVaR risk. We set the parameter values exactly the same as those in Model (M3) and run Model (M4). The maximum expected mean return obtained is 111.13% and the corresponding worst-case return is 92.75%. This pair of worst-case/mean return ratios given another extreme point on the efficient frontier.

To generate the whole frontier, we add the following constraint to Model (M4)

$$\text{CVaR}_\beta(-r^T x) \leq -\omega \tag{23}$$

We vary $\omega$ from 92.75% to 100.52% and solve the corresponding Model (M4) to get a point on the frontier. The resulting efficient frontier is shown in Figure 2. The computation of the whole efficient frontier involves invoking IEM on a sequence of 104 instances, and the exercise took 35 minutes on a Lenovo Thinkpad T60p, with an Intel CPU T2600 at 2.16 GHz, and 2 GB RAM.

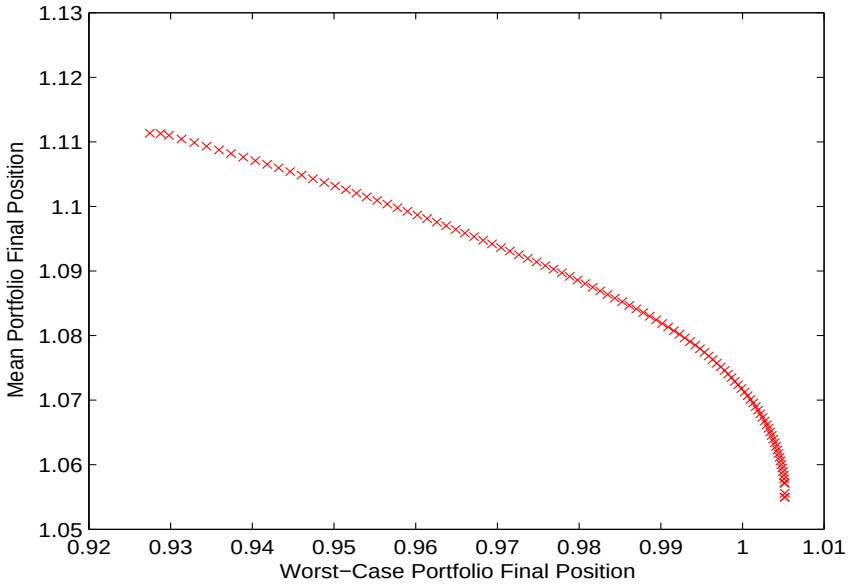


Figure 2: Efficient Frontier of the portfolio final position

## 6 Conclusion

IEM is an efficient algorithm for stochastic linear programs with CVaR constraints. A fundamental characterization of coherent risk measures in general, and CVaR in specific, is that they can be

represented as the worst-case expectation in some probability space. IEM exploits this property to sequentially approximate CVaR constraints and solve CVaR-constrained stochastic linear problems simultaneously. It is easy to implement and enjoys computational advantages in terms of efficiency over the monolithic method proposed in [7].

## References


[1] Philippe Artzner, Freddy Delbaen, Jean-Marc Eber, and David Heath. Coherent measures of risk. *Mathematical Finance*, 9:203–228, 1999.

[2] P. Krokhmal, J. Palmquis, and S. Uryasev. Portfolio optimization with conditional value-at-risk objective and constraints. *Journal of Risk*, 4(2):11–27, 2002.

[3] A. Künzi-Bay and János Mayer. Computational aspects of minimizing conditional value-at-risk. *Computational Management Science*, 3:3–27, 2006.

[4] B. John Manistre and Geoffrey H. Hancock. Variance of the cte estimator. *North American Actuarial Journal*, 9:129–156, April 2005.

[5] H. N. Nagaraja. Some nondegenerate limit laws for the selection differential. *The Annals of Statistics*, 10:1306–1310, 1982.

[6] Arkadi Nemirovski and Alexander Shapiro. Convex approximations of chance constrained programs. *SIAM Journal on Optimization*, 17:969–996, 2006.

[7] R. Tyrrell Rockafellar and Stanislav Uryasev. Optimization of conditional value-at-risk. *The Journal of Risk*, 2:21–41, 2000.

[8] R. Tyrrell Rockafellar and Stanislav Uryasev. Conditional value-at-risk for general loss distributions. *Technical Report, University of Florida*, 2001.