CMS 3: 3–27 (2006)

DOI: 10.1007/s10287-005-0042-0

Computational aspects of minimizing conditional value-at-risk

Alexandra Künzi–Bay, János Mayer*

Institute for Operations Research, University of Zurich, Moussonstrasse 15, 8044 Zurich, Switzerland (e-mail: kuenzi@ior.unizh.ch)

Abstract. We consider optimization problems for minimizing conditional value–at–risk (CVaR) from a computational point of view, with an emphasis on financial applications. As a general solution approach, we suggest to reformulate these CVaR optimization problems as two–stage recourse problems of stochastic programming. Specializing the L–shaped method leads to a new algorithm for minimizing conditional value–at–risk. We implemented the algorithm as the solver *CVaRMin*. For illustrating the performance of this algorithm, we present some comparative computational results with two kinds of test problems. Firstly, we consider portfolio optimization problems with 5 random variables. Such problems involving conditional value at risk play an important role in financial risk management. Therefore, besides testing the performance of the proposed algorithm, we also present computational results of interest in finance. Secondly, with the explicit aim of testing algorithm performance, we also present comparative computational results with randomly generated test problems involving 50 random variables. In all our tests, the experimental solver, based on the new approach, outperformed by at least one order of magnitude all general–purpose solvers, with an accuracy of solution being in the same range as that with the LP solvers.

Keywords: Conditional value–at–risk; Stochastic programming; Mathematical programming algorithms; Stochastic models; Finance; Portfolio optimization; Risk management

* Financial support by the national center of competence in research "Financial Valuation and Risk Management" is gratefully acknowledged. The national centers in research are managed by the Swiss National Science Foundation on behalf of the federal authorities.

CMS Computational Management Science

1 Introduction

In the financial industry, value–at–risk (VaR) is a widely used concept for quantifying the downside risk of portfolios. A major drawback of this approach is that optimization problems, aiming at computing optimal portfolios with respect to VaR, are typically hard to solve numerically. The reason is that VaR is in general not a convex function of the portfolio weights. A related concept, conditional value–at–risk (CVaR), has recently been suggested as an alternative downside risk measure. This concept enjoys increasing interest in the finance industry. One of the reasons is that, unlike VaR, the computation of CVaR–optimal portfolios leads to convex programming problems. For a finite discrete distribution, the situation is even better: optimal portfolios can be computed by solving linear programming (LP) problems. The currently pursued approach is to solve those LP problems by general–purpose LP solvers.

In this paper, we consider optimization problems aiming at minimizing CVaR and focus on the computational point of view. The main idea is as follows: the single–stage CVaR minimization problems can be reformulated as two–stage recourse problems of stochastic programming. This way we can apply the techniques available for two–stage recourse problems for solving the CVaR minimization problems. Specializing the L–shaped method leads to a new algorithm for minimizing conditional value–at–risk. According to our initial computational experience, the new method turns out to be quite efficient.

We have developed an experimental implementation of the algorithm as the solver *CVaRMin*. For illustrating the computational performance of *CVaRMin*, we present comparative numerical results obtained by solving two kinds of test problems: portfolio optimization problems involving the minimization of CVaR and randomly generated test problems. For portfolio optimization, we also present some parametric computational results. The computations have been carried out by using the model management system SLP–IOR for stochastic linear programming.

2 Conditional value-at-risk

We consider in general the random variable

$$\zeta(x, \eta, \xi) := \eta^{\mathrm{T}} x - \xi \tag{1}$$

where $x \in \mathbb{R}^n$ is a vector of decision variables, η is an n–dimensional random vector, and ξ is a random variable. (η, ξ) is defined on a probability space $(\Omega, \mathcal{F}, P)$. We assume that (η, ξ) has a finite expected value and introduce the notation $\bar{\eta} = \mathbb{E}[\eta]$ and $\bar{\xi} = \mathbb{E}[\xi]$. Positive values of $\zeta(x, \eta, \xi)$ will be interpreted as losses and negative values as gains. In the sequel, we will concentrate on losses; therefore gains will be viewed as negative losses. The probability distribution function of $\zeta(x, \eta, \xi)$ will be denoted by $\Psi(x, \cdot)$ and will be called the *profit–loss distribution function*. The

quantile function $v : \mathbb{R}^n \times (0, 1) \to \mathbb{R}$ associated with $\zeta(x, \eta, \xi)$ will be defined as

$$\begin{aligned} v(x, \alpha) &:= \min \{ y \mid \Psi(x, y) \geq \alpha \} \\ &= \min \{ y \mid \mathbb{P}[\zeta(x, \eta, \xi) \leq y] \geq \alpha \}. \end{aligned} \tag{2}$$

In mathematical terms, for a fixed x and $\alpha \in (0, 1)$ the corresponding value of the quantile function is the left endpoint of the closed interval of the α–quantiles of $\zeta(x, \eta, \xi)$. The interpretation in terms of losses is the following: $v(x, \alpha)$ is the minimal loss level such that with probability α the loss will not exceed $v(x, \alpha)$. In this context, α will be chosen as a large probability level, for instance, $\alpha = 0.95$. In optimization problems, the goal is to minimize $v(x, \alpha)$ or to keep its value below a prescribed level.

In financial applications, $v(x, \alpha)$ is called *value-at-risk (VaR)* and is widely used as a risk measure for evaluating $\zeta(x, \eta, \xi)$; for a discussion of VaR as a risk measure see, for instance, Elton, Gruber, Brown, and Goetzmann (2003) or Jorion (1996). Unfortunately, VaR is not a coherent risk measure as discussed by Artzner, Delbaen, Eber, and Heath (1999). From the optimization point of view, the most important implication is that, apart from some special cases, $v(x, \alpha)$ is not a convex function. Consequently, optimization problems aiming at minimizing $v(x, \alpha)$ are in general non–convex problems and therefore difficult to solve numerically.

Another obvious drawback is the following: although with a high probability α the loss will not exceed $v(x, \alpha)$, this measure does not account for the loss–size concerning events when the loss exceeds $v(x, \alpha)$. Motivated by this shortcoming, Rockafellar and Uryasev (2000) introduced the following risk measure for continuous distribution functions

$$v_C(x, \alpha) := \mathbb{E}[\zeta(x, \eta, \xi) | \zeta(x, \eta, \xi) \geq v(x, \alpha)] \tag{3}$$

with $v_C(x, \alpha)$ as the conditional expected loss given that the loss exceeds $v(x, \alpha)$. The function $v_C(x, \alpha)$ is called *conditional value-at-risk (CVaR)* function. Rockafellar and Uryasev have shown that, considered as a risk measure, v_C is coherent, and have derived the following representation

$$v_C(x, \alpha) = \min_z \left[z + \frac{1}{1 - \alpha} \mathbb{E}[(\zeta(x, \eta, \xi) - z)^+] \right] \tag{4}$$

where $\mathsf{y}^+ = \max\{\mathsf{y}, 0\}$ is the positive part of $\mathsf{y} \in \mathbb{R}$. The set of optimal solutions of (4) coincides with the interval of the α–quantiles of $\zeta(x, \eta, \xi)$. The representation (4) plays a crucial role from the optimization point of view. On the one hand, it immediately implies that $v_C(x, \alpha)$ is a convex function of x. On the other hand, it is the basis of algorithms for the solution of the corresponding optimization problems. Consequently, this risk measure is well–suited for building optimization problems aiming at minimizing CVaR.

For general distributions and CVaR defined according to (3), the above representation does not hold. According to an idea of Pflug (2000), the representation

(4) will be taken as the *definition* of CVaR. For general distributions, Rockafellar and Uryasev (2002) have shown that CVaR, defined according to (4), has an interpretation in terms of conditional expectation, if the conditional expectation is taken according to a tail distribution. Acerbi (2002) gave a representation in terms of an average over α of the VaR values $v(x, \alpha)$. For detailed discussions of the properties of CVaR see the papers of Rockafellar and Uryasev (2002), Acerbi (2002) and Acerbi and Tasche (2002).

3 Minimizing CVaR

In this paper, we will concentrate on optimization problems with CVaR in the objective; for problems involving CVaR constraints see Krokhmal, Palmquist, and Uryasev (2002). We consider the following prototype problem

$$\left.\begin{array}{ll} \min\limits_{x} & c^{\mathrm{T}}x + v_C(x, \alpha) \\ \text{s.t.} & x \in \mathcal{P} \end{array}\right\} \tag{5}$$

where $\mathcal{P}$ is a polyhedral set given, for example, in standard form $\mathcal{P} = \{x \mid Ax = b,\ x \geq 0\}$ with A being an $m \times n$ matrix and $b \in \mathbb{R}^m$. For the sake of simplicity, we assume that $\mathcal{P} \neq \emptyset$ and that $\mathcal{P}$ is bounded. Substituting the definition of $v_C(x, \alpha)$ from (4), we get the following optimization problem (Rockafellar and Uryasev (2000))

$$\left.\begin{array}{ll} \min\limits_{x,z} & c^{\mathrm{T}}x + z + \frac{1}{1-\alpha}\,\mathbb{E}[(\eta^{\mathrm{T}}x - \xi - z)^+] \\ \text{s.t.} & x \in \mathcal{P}. \end{array}\right\} \tag{6}$$

This is a linearly constrained convex programming problem. We observe that (6) can be equivalently formulated as the following two–stage recourse problem

$$\left.\begin{array}{ll} \min\limits_{x,z} & c^{\mathrm{T}}x + z + \mathbb{E}[Q_C(x, z, \eta, \xi)] \\ \text{s.t.} & x \in \mathcal{P} \end{array}\right\} \tag{7}$$

with the recourse subproblem

$$\left.\begin{array}{rll} Q_C(x, z, \eta, \xi) := \frac{1}{1-\alpha} & \min\limits_{y} & y \\ & \text{s.t.} & y \geq \eta^{\mathrm{T}}x - \xi - z \\ & & y \geq 0. \end{array}\right\} \tag{8}$$

The LP dual of the recourse subproblem (8) is

$$\left.\begin{array}{rll} Q_C(x, z, \eta, \xi) = \frac{1}{1-\alpha} & \max\limits_{u} & (\eta^{\mathrm{T}}x - \xi - z)u \\ & \text{s.t.} & 0 \leq u \leq 1, \end{array}\right\} \tag{9}$$

which has an optimal solution $u = 0$ or $u = 1$, depending on the sign of $\eta^{\mathrm{T}} x - \xi - z$, and the optimal objective value is $\frac{1}{1-\alpha}(\eta^{\mathrm{T}} x - \xi - z)^+$. This shows the equivalence of (6) and (7). For a discussion of two–stage recourse problems see, for instance, Birge and Louveaux (1997), Kall and Mayer (2005b), or Kall and Wallace (1994).

The reformulation (7) has the following algorithmic implication: solution methods, designed for two–stage recourse problems, can be considered as candidates for solving the CVaR minimization problem (6). In particular, there are algorithms available for solving (7) with continuously distributed random vectors (η, ξ). Examples of algorithms of this type are the successive discrete approximation method or the sample average approximation method (SAA); these methods are discussed, for instance, in Kall and Mayer (2005b), Kall and Wallace (1994), and Linderoth, Shapiro, and Wright (2005). Note that the algorithm suggested by Rockafellar and Uryasev (2000) for the solution of (6) is, from the two–stage recourse point of view, essentially the SAA method.

Notice that (7) has a simple recourse structure (the recourse matrix is a simple recourse matrix), but besides the right–hand–side, the technology matrix may also be stochastic. If in (7) only the right–hand–side is stochastic, that is, if $\eta \equiv t$ with $t \in \mathbb{R}^n$ holds, then we have a simple recourse problem in the classical sense. For such problems quite efficient algorithms exists, see Kall and Mayer (2005b) and Kall and Wallace (1994), and for computational results see Kall and Mayer (2005a). In the general case, when η is random vector with a non–degenerate distribution, the traditional way for solving such problems in stochastic programming consisted in applying methods, designed for complete recourse problems, like the methods mentioned in the previous paragraph.

Recently Klein Haneveld and Van der Vlerk (2002) proposed an algorithm for general simple recourse problems with a random technology matrix, which we consider an important development in stochastic programming. Starting with algorithms for integrated chance constraints, the authors arrive at a cutting–plane algorithm for general problems with simple recourse structure. For the case of discrete distributions, their method is essentially the Benders algorithm, specialized to the structure. In this paper, we start with the two–stage recourse formulation (7) of CVaR minimization problems and design a specialized version of Benders decomposition. From the purely mathematical point of view, our proposed method can be considered as a version of the general method of Klein Haneveld and Van der Vlerk, as specialized to CVaR minimization.

In the sequel, we will solely consider the case where the probability distribution of $\zeta(x, \eta, \xi)$ is finite discrete; the probability distribution is then given by the tableau

$$\begin{pmatrix} p_1 & \dots & p_N \\ \hat{\eta}^1 & \dots & \hat{\eta}^N \\ \hat{\xi}^1 & \dots & \hat{\xi}^N \end{pmatrix} \tag{10}$$

where $(\hat{\eta}^k, \hat{\xi}^k)$ is the k^{th} realization of (η, ξ) with the corresponding probability $p_k > 0,\ \forall\, k = 1, \ldots, N$, and $\sum_{k=1}^{N} p_k = 1$ holds.

It is well–known and easy to see that in the discretely distributed case the two–stage recourse problem (7) can be equivalently written as the following linear programming problem

$$\left.\begin{array}{lll} \min\limits_{x,z} & c^{\mathrm{T}}x + z + \frac{1}{1-\alpha} \sum\limits_{k=1}^{N} p_k y_k & \\ \text{s.t.} & (\hat{\eta}^k)^{\mathrm{T}}x - z - y_k \leq \hat{\xi}^k,\ k = 1, \ldots, N & \\ & y_k \geq 0,\ k = 1, \ldots, N & \\ & x \in \mathcal{P}. & \end{array}\right\} \tag{11}$$

The above LP formulation of the CVaR minimization problem, via the representation (4), has been proposed by Rockafellar and Uryasev (2000) as a basis for numerical solution. This result is a breakthrough, regarding the numerical solution of optimization problems involving CVaR. In their numerical experiments, the authors use a general–purpose LP solver for solving (11).

We will explore how the special structure of the problem can be utilized for building an algorithm for our problem. The starting point is the L–shaped method for two–stage recourse problems with finite discrete distribution, which is one of the most widely used technique for this problem class; see, for instance, Birge and Louveaux (1997), Kall and Mayer (2005b), or Kall and Wallace (1994). The L–shaped algorithm is based on the Benders decomposition method (Benders (1962)), which has been specialized for the structure of two–stage recourse problems by Van Slyke and Wets (1969). We propose an algorithm for the solution of the CVaR minimization problem (6) by further specializing the L–shaped method for the structure of the equivalent two–stage recourse problem (7).

The main idea of the L–shaped method is a reformulation of (7) in terms of optimal solutions u^k of the dual recourse–subproblems (9) corresponding to the realizations $(\hat{\eta}^k, \hat{\xi}^k)$, $k = 1, \ldots, N$. As we have seen above, the dual problem (9) is extremely simple: the dual–feasible vectors are just scalars, and the optimal solution of the dual problem is either $u^k = 0$ or $u^k = 1$. With the L–shaped method, the (aggregate) cuts are constructed on the basis of the N–dimensional vector $(u^1, \ldots, u^N)^{\mathrm{T}}$. In our case, this is a binary vector, which can be identified in a one–to–one manner with a subset of the index–set $\mathcal{N} = \{1, \ldots, N\}$; this identification is done by choosing those indices k as elements for which $u^k = 1$ holds. Consequently, the equivalent formulation of (7) in terms of Benders cuts

assumes the following form

$$\left.\begin{array}{ll} \min\limits_{x,z,w} & c^{\mathrm{T}}x + z + \frac{1}{1-\alpha}\, w \\ \text{s.t.} & \sum\limits_{k\in\mathcal{K}} p_k \left((\hat{\eta}^k)^{\mathrm{T}} x - \hat{\xi}^k - z \right) - w \le 0, \quad \mathcal{K} \subset \mathcal{N} \\ & x \qquad\qquad\qquad\quad \in \mathcal{P} \end{array}\right\} \tag{12}$$

which will be called the *full master problem*. Note that in (12) we have 2^N inequalities out of which the inequality corresponding to $\mathcal{K} = \emptyset$ just requires the non–negativity of w. For large N this results in master problems having an astronomical number of inequality constraints. Taking $N = 100$, for instance, we have no chance to set up (12) for numerical solution nor to solve it directly by a general–purpose LP solver. Notice that, in general, many of the constraints in (12) may be redundant; due to the constraint $x \in \mathcal{P}$, not all of the N–dimensional binary vectors will appear in the underlying duality consideration.

The idea of the L–shaped method is constraint–generation: the constraints are included into (12) in a successive manner. In each step the current *relaxed master problem*

$$\left.\begin{array}{lll} \underline{F}_\nu := \min\limits_{x,z,w} & c^{\mathrm{T}}x + z + \frac{1}{1-\alpha}\, w & \\ \text{s.t.} & \sum\limits_{k\in\mathcal{K}_i} p_k \left((\hat{\eta}^k)^{\mathrm{T}} x - \hat{\xi}^k - z \right) - w \le 0, & i = 1, \ldots, \nu \\ & w \ge 0 & \\ & x \qquad\qquad \in \mathcal{P} & \end{array}\right\} \tag{13}$$

is solved, where ν is the number of constraints generated so far, and $\mathcal{K}_i \subset \mathcal{N}\ \forall\, i$, $\mathcal{K}_i \neq \mathcal{K}_l$ for $i \neq l$. It is easy to see that under our assumptions this problem has an optimal solution. Based on the solution of (13), the next constraint is added (a cut is generated).

Before proceeding with the formal specification of the algorithm, let us pause for a moment for discussing some implications of the equivalent representation (12). Let

$$\mathcal{D} := \left\{ (x, z, w) \mid \sum_{k=1}^{N} p_k \left(\zeta(x, \hat{\eta}^k, \hat{\xi}^k) - z \right)^+ - w \le 0 \right\}.$$

We have the following polyhedral representation for this set:

Proposition 1

$$\mathcal{D} = \bigcap_{\mathcal{K}\subset\mathcal{N}} \left\{ (x, z, w) \mid \sum_{k\in\mathcal{K}} p_k \left((\hat{\eta}^k)^{\mathrm{T}} x - \hat{\xi}^k - z \right) - w \le 0 \right\} \tag{14}$$

with the sum defined as zero for $\mathcal{K} = \emptyset$.

Proof. The proposition follows immediately from the equivalent representation (12). An alternative, direct proof is based on the following fact: for arbitrary real numbers $\alpha_1, \ldots, \alpha_N$ the equality

$$\sum_{k=1}^{N} \alpha_k^+ = \max_{\mathcal{K} \in \mathcal{N}} \sum_{k \in \mathcal{K}} \alpha_k$$

obviously holds, and the maximum on the right–hand–side is attained for the set $\mathcal{K}^* = \{k \mid \alpha_k > 0\}$. This implies the proposition. □

Proposition 1 is the CVaR–analogue of a polyhedral representation result concerning integrated chance constraints of Klein Haneveld and Van der Vlerk (2002). The direct proof given above follows the same lines as the proof in the cited paper.

The formal specification of the L–shaped method, as specialized for (6) with a finite discrete distribution, follows. We introduce the notation

$$\hat{\eta}_{[i]} := \sum_{k \in \mathcal{K}_i} p_k \hat{\eta}^k, \quad \hat{\xi}_{[i]} := \sum_{k \in \mathcal{K}_i} p_k \hat{\xi}^k, \quad p_{[i]} := \sum_{k \in \mathcal{K}_i} p_k, \tag{15}$$

and utilize this to formulate the relaxed master problem (13) as

$$\left.\begin{array}{lllll} \min\limits_{x,z,w} & c^{\mathrm{T}}x + z + \frac{1}{1-\alpha}\, w & & & \\ \text{s.t.} & \hat{\eta}_{[i]}^{\mathrm{T}} x & -p_{[i]} z & -w \le \hat{\xi}_{[i]}, & i = 1, \ldots, \nu \\ & & & w \ge 0 & \\ & x & & \in \mathcal{P}. & \end{array}\right\} \tag{16}$$

The algorithm runs as follows.

Step 0. (Initialize)
Let $\mathcal{K}_1 := \mathcal{N}$ and consequently $\hat{\eta}_{[1]} = \bar{\eta}$, $\hat{\xi}_{[1]} = \bar{\xi}$, and $p_{[1]} = 1$. Set $\nu := 1$. The single inequality constraint in the relaxed master problem will be $\bar{\eta}^{\mathrm{T}} x - z - w \le \bar{\xi}$.

Step 1. (Solve the relaxed master problem)
Solve the current relaxed master problem (16). Let (x^*, z^*, w^*) be an optimal solution, and let
$\mathcal{K}^* := \{k \mid 1 \le k \le N,\ (\hat{\eta}^k)^{\mathrm{T}} x^* - \hat{\xi}^k - z^* > 0\}$ and
$w_+^* := \sum\limits_{k \in \mathcal{K}^*} p_k \left((\hat{\eta}^k)^{\mathrm{T}} x^* - \hat{\xi}^k - z^* \right)$.

Step 2. (Check for optimality)
If $w_+^* - w^* \le 0$ then **Stop**; x^* is an optimal solution of (6). Otherwise continue with the next step.

Step 3. (Append a cut to the relaxed master problem)
Set $\nu := \nu + 1$, $\mathcal{K}_\nu = \mathcal{K}^*$, and compute $\hat{\eta}_{[\nu]}$, $\hat{\xi}_{[\nu]}$ and $p_{[\nu]}$ according to (15). Append the corresponding cut to the set of constraints in the relaxed master problem (16). **Continue with *Step 1*.**

For the sake of completeness, we state the well-known finiteness result concerning Benders decomposition and give a direct proof for it in our special case.

Proposition 2 *The above algorithm finds an optimal solution of (6) in a finite number of iterations.*

Proof. If in *Step 1* $\mathcal{K}^* = \mathcal{K}_i$ holds for some $1 \leq i \leq \nu$, then the stopping criterion holds, and the algorithm stops in *Step 2*. Thus, none of the subsets $\mathcal{K}_i \subset \mathcal{N}$ is repeated, and consequently the algorithm stops after a finite number of iterations. The optimal objective value $\underline{F}_\nu := c^{\mathrm{T}}x^* + z^* + \frac{1}{1-\alpha}\, w^*$ of the relaxed master problem (13) is obviously a lower bound for the optimal objective value of (6). On the other hand, we have the inequality

$$\sum_{k\in\mathcal{K}} p_k \left((\hat{\eta}^k)^{\mathrm{T}}x^* - \hat{\xi}^k - z^*\right) \leq \sum_{k\in\mathcal{K}^*} p_k \left((\hat{\eta}^k)^{\mathrm{T}}x^* - \hat{\xi}^k - z^*\right) = w_+^*$$

which holds for any $\mathcal{K} \subset \mathcal{N}$, due to the definition of $\mathcal{K}^*$. Thus, (x^*, z^*, w_+^*) is a feasible solution of the full master problem (12). Consequently, $\overline{F}_\nu := c^{\mathrm{T}}x^* + z^* + \frac{1}{1-\alpha}\, w_+^*$ is an upper bound. Finally, the stopping criterion obviously implies the inequality $\overline{F}_\nu - \underline{F}_\nu \leq 0$ which is equivalent to $\overline{F}_\nu = \underline{F}_\nu$, thus completing the proof of the proposition. □

From the purely mathematical point of view, our algorithm is the Benders decomposition method applied to (7). From the computational point of view, the L–shaped method has been adapted to the special structure of (7). As mentioned above, the two–stage equivalent (7) has a simple recourse structure with a random technology matrix, for which Klein Haneveld and Van der Vlerk (2002) proposed an algorithm for the finitely distributed case. From this viewpoint, our algorithm may be also considered as a variant of the method of Klein Haneveld and Van der Vlerk, as applied to the CVaR minimization problem.

Whereas the lower bounds $\underline{F}_\nu$ are monotonically increasing, the upper bounds $\overline{F}_\nu$ do not form a monotonically decreasing sequence. Therefore, instead of the stopping criterion $\overline{F}_\nu - \underline{F}_\nu \leq 0$, it is reasonable to use for $\nu > 1$ the stopping criterion $\overline{F}_\nu^* - \underline{F}_\nu \leq 0$, where $\overline{F}_\nu^*$ is the best (lowest) upper bound found in the iterations so far. Formally, the definition is $\overline{F}_1^* := \overline{F}_1$ and for $\nu > 1$, $\overline{F}_\nu^* := \min\{\overline{F}_\nu, \overline{F}_{\nu-1}^*\}$. This observation is due to Benders (1962). It is easy to see that the modified algorithm is still finite. This modification becomes relevant in the implementation, where the stopping rule is adapted to the floating point arithmetic by employing an $\varepsilon > 0$ tolerance, see the stopping rule (18) in the next section. In the basic algorithm, as formulated above, the algorithm will stop at the next

iteration, if by solving the relaxed master problem an optimal solution of (12) is obtained.

Finally, let us remark that the polyhedral representation above and the algorithm presented carry over, with obvious changes, to the case of problems with CVaR constraints.

4 Computational results

We utilized the model management system SLP–IOR as a workbench for our computational experiments. For a detailed description of the architecture and the features of SLP–IOR see, for example, Kall and Mayer (1996), Kall and Mayer (2004b), Kall and Mayer (2005c), and references therein. For the solvers connected to SLP–IOR also see Mayer (1998). Let us emphasize here one of the features of SLP–IOR: it has an interface to the solvers available with the general algebraic modeling language GAMS (Brooke, Kendrick, Meeraus, and Raman (1998); some of those solvers will participate in our experiments). The computations were carried out on a 2.6 GHz Pentium-III PC with 1 GB RAM, under the operating system Windows 2000.

For the specialized version of the L–shaped method, as described in Section 3, we implemented a first experimental solver, called *CVaRMin*. The solver has been developed in Delphi 7.0.

For solving LP subproblems in *CVaRMin*, Minos 5.4 (Murtagh and Saunders (1978)) has been employed. Minos is a commercial solver, available in source form, and is primarily aimed at NLP problems. In our algorithm, we have to solve a sequence of relaxed master LP problems. In this first implementation, the primal form (13) is solved. However, in the sequence of LP's, apart from the first one, each LP differs from its predecessor by a single additional row, corresponding to the cut. Therefore, a straightforward idea is to solve the dual of (13), which then has an additional column. Thus, the optimal basis of the dual problem from the predecessor can be employed as a starting basis in a hot start, which most probably results in reduced computational time. This will be the next step in the future development of the solver.

Implementing the algorithm, due to the finite precision arithmetic, the stopping criterion in *Step 2* of the method has to be modified by employing a stopping tolerance ε. A straightforward stopping criterion is

$$\overline{F}_\nu - \underline{F}_\nu = \frac{1}{1-\alpha}\,(w_+^* - w^*) \le \varepsilon, \tag{17}$$

that is, the algorithm is stopped, when the gap between upper– and lower– bounds becomes smaller than the prescribed tolerance ε. In most practical cases, the relative error is what matters, therefore we have implemented in *CVaRMin* a stopping rule involving the relative error. Furthermore, instead of the current upper bound, at iteration $\nu > 1$ the best upper bound $\overline{F}_\nu^*$ found so far is used. The implemented

stopping rule is

$$\frac{\overline{F}_\nu^* - \underline{F}_\nu}{|\underline{F}_\nu|} \leq \varepsilon \tag{18}$$

for $|\underline{F}_\nu| > 10^{-8}$, and $\overline{F}_\nu^* - \underline{F}_\nu \leq \varepsilon$ otherwise. In our computations, the stopping tolerance has been chosen as $\varepsilon = 10^{-10}$.

In the experiments, besides *CVaRMin*, we employed several solvers for comparative purposes. On the one hand, we chose the GAMS general–purpose LP solvers *GAMS/Cplex* 9.0 and *GAMS/OSL* Version 1, see Gams Development Corporation (2004), with the default setting of the run–time–parameters. The nonlinear programming solvers *GAMS/Conopt*, *GAMS/Minos*, and *GAMS/Snopt* were also employed. Additionally, we took *BPMPD* 2.1, an interior point LP solver, developed by Mészáros (1997), with stopping tolerance 10^{-10} for the relative duality gap. Although *BPMPD* is a general–purpose LP solver, it turned out that it is especially well–suited for solving two–stage recourse problems with a finite discrete distribution, see Mészáros (1997) and Kall and Mayer (2005a). We employed also *QDECOM* (1985), developed by Ruszczyński (1986), a regularized version of the L–shaped method. *QDECOM* is, according to our computational experience with two–stage recourse problems in general, an excellent solver for this type of problems. In the meantime, a significantly improved version and implementation in the solver *Decomp* has been developed by Ruszczyński and Świętanowski (1997). However, this solver is currently not connected to SLP–IOR, for technical reasons.

Concerning the computational experiments, two kinds of test problems were employed. First, we applied the algorithm for solving portfolio optimization problems of the CVaR minimization type, similarly as the authors did in Rockafellar and Uryasev (2000). The difference lies in the choice of the asset classes. Concerning the algorithm, this gives a first impression on the performance of the method for small–scale problems involving 5 random variables. On the other hand, this way we also provide an example for the practical application of the abstract problem type (5). For readers interested in portfolio optimization, we have also included some parametric results. These computations also serve for illustrating the effect of a warm starting strategy concerning the method. Secondly, we solved randomly generated test problems with 50 random variables, with the sole purpose of illustrating the comparative computational performance of the algorithm and the accuracy of the results.

Interested readers may perform their own experiments: SLP–IOR is freely available for academic purposes, with the solvers *CVaRMin*, *BPMPD*, and *QDECOM* connected to it; for obtaining a copy please contact Janos Mayer (e-mail: mayer@ior.unizh.ch). Concerning the commercial solvers *GAMS/Cplex* and *GAMS/OSL*, however, a separate licence for GAMS and for these solvers is needed. The test problems used in the computations are also available on request, in SLP–IOR input format.

4.1 Portfolio optimization

In this section, we present computational results concerning financial portfolio optimization. Let us consider a one–stage portfolio selection problem, which belongs to the class of problems discussed in Section 3. It corresponds to the classical standard–problem of Markowitz (1959) with the variance as a risk measure replaced by CVaR. Portfolio optimization problems of this kind have firstly been formulated by Krokhmal, Palmquist, and Uryasev (2002). We have n risky assets with random returns $\tilde{r}_1, \ldots, \tilde{r}_n$. Denoting the portfolio weights by $x_1, \ldots, x_n$, we consider the following problem of the type (5):

$$\left.\begin{array}{l} \min\ -\bar{r}^{\mathrm{T}}x + v_C(x,\alpha) \\ \text{s.t.}\ \displaystyle\sum_{i=1}^{n} x_i = 1 \\ \qquad x_i \geq 0, \quad i = 1, \ldots, n, \end{array}\right\} \tag{19}$$

where $\bar{r}$ is the vector of expected returns, that means, $\bar{r} = \mathbb{E}[\tilde{r}]$ holds. $v_C(x,\alpha)$ is the CVaR value corresponding to the loss function $\zeta(x,\tilde{r}) := -\tilde{r}^{\mathrm{T}}x$; the loss that enters the definition of CVaR is the negative portfolio return.

We consider portfolios consisting of five risky asset classes: Swiss equities, European equities, world equities, Swiss franc bonds, and global bonds. To describe the properties of these asset classes, we use the price data of the following total return indices that are all provided by Datastream®: MSCI Switzerland (in the sequel abbreviated to MSCI.CH), MSCI Europe ex Switzerland denominated in Swiss francs (MSCI.E), MSCI World ex Europe denominated in Swiss francs (MSCI.W), Pictet General Bond Total (Pictet.Bond), and J.P. Morgan Global Broad Index denominated in Swiss francs (JPM.Global). The original monthly price data of the indices range from January 1987 till December 2002. Using S-PLUS® 6.1 for Windows, Professional Edition, Release 1 and the functions in S+FinMetrics, we calculated the monthly continuously compounded returns ranging from February 1987 till December 2002; that gives a total number of 191 observations. Figure 1 displays the returns for two of the indices. The horizontal lines in the figure indicate the zero–return level.

All the following statistical analyzes were also done using S-PLUS and S+FinMetrics; for analyzing financial time series data with S-PLUS® and with the functions in S+FinMetrics, see, for example, Zivot and Wang (2003). The mean $\bar{r}$ and the covariance matrix Σ of the monthly returns are given in Table 1 and Table 2, respectively.

Assuming a joint normal distribution for the asset returns, we generated the realizations of the returns by random sampling from a multivariate normal distribution with the parameters $\bar{r}$ and Σ. The pseudo–random sequences were generated by S-PLUS. The straightforward method has been used, which consists of generating standard normal variates first, followed by a linear transformation with the lower–

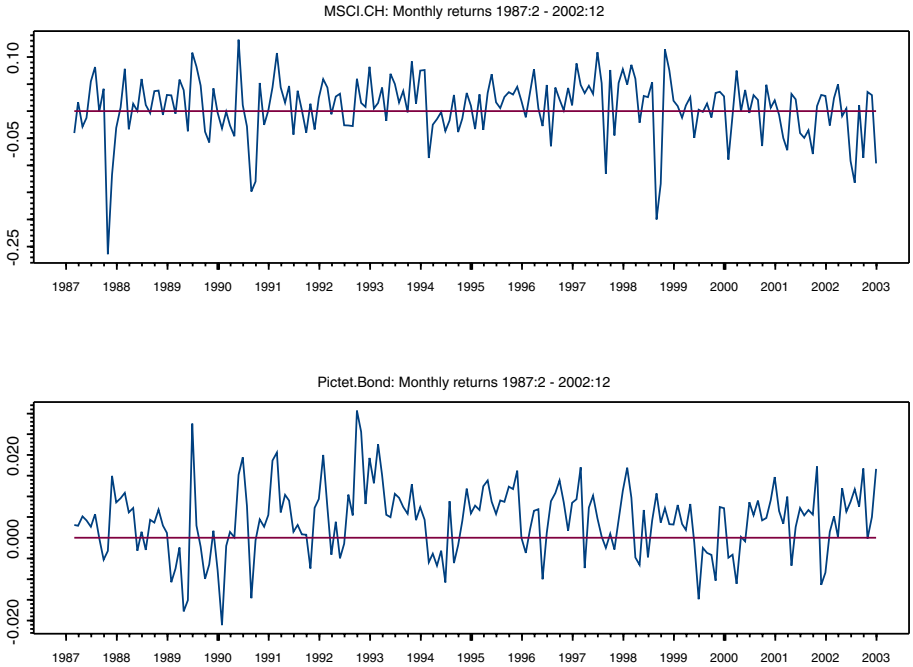

Fig. 1. Return data for the MSCI.CH and Pictet.Bond indices

Table 1. Mean of the monthly returns

Index	mean return
MSCI.CH	0.007417
MSCI.E	0.005822
MSCI.W	0.004236
Pictet.Bond	0.004231
JPM.Global	0.005534

Table 2. Covariance matrix of the monthly returns

Index	MSCI.CH	MSCI.E	MSCI.W	Pictet.Bond	JPM.Global
MSCI.CH	0.003059	0.002556	0.002327	0.000095	0.000533
MSCI.E	0.002556	0.003384	0.002929	0.000032	0.000762
MSCI.W	0.002327	0.002929	0.003509	0.000036	0.000908
Pictet.Bond	0.000095	0.000032	0.000036	0.000069	0.000048
JPM.Global	0.000533	0.000762	0.000908	0.000048	0.000564

Table 3. Test runs #1. Elapsed time summary (sec)

N	BPMPD	Cplex S	Cplex IP	CVaRMin	OSL1 S	QDECOM
500	0.03	0.05	0.09	0.01	0.23	0.19
1000	0.08	0.13	0.09	0.01	0.72	0.34
3000	0.30	0.69	0.59	0.02	6.40	4.34
5000	0.45	2.52	1.45	0.05	17.88	8.70
10000	1.17	12.31	5.86	0.08	74.59	41.62
20000	2.42	55.23	30.33	0.19	273.13	137.22

triangular Cholesky–factor of Σ. For the details of this method see, for instance, Ripley (1987).

Thus, each realization $\hat{r}_k$, $k = 1, \ldots, N$ has an equal probability $p_k = \frac{1}{N}$ to occur. This is clearly not a highly sophisticated scenario generation method and is not suitable for prediction purposes. For the computational experiments, the sample size N is chosen to be 500, 1'000, 3'000, 5'000, 10'000, and 20'000.

Let us note that, under our assumption of having a multivariate normal distribution, the CVaR term in the objective function can be computed as

$$v_C(x, \alpha) = -\bar{r}^{\mathrm{T}} x + \gamma \cdot \sqrt{x^{\mathrm{T}} \Sigma x}, \tag{20}$$

with an appropriately chosen constant γ, see Rockafellar and Uryasev (2000). Thus, (19) can also be directly solved by general–purpose nonlinear programming (NLP) solvers.

Test runs #1. Portfolio optimization: base case

The test problem battery for these tests consists of 7 test problems. By random sampling, we generated 6 test problems, corresponding to the sample sizes $N =$ 500, 1′000, 3′000, 5′000, 10′000, and 20′000. Besides these, we also took the nonlinear programming formulation according to (20). The reliability level was chosen as $\alpha = 0.9$.

First, we solved the nonlinear programming formulation of (19), with the CVaR objective function term (20). For this we used *GAMS/Conopt*, *GAMS/Minos*, and *GAMS/Snopt*. The optimal objective value turned out to be 0.0058959347 with $v_C(x^*, \alpha) = 0.0102071578$ and $z^* = 0.0062906177$, where (x^*, z^*) denotes an optimal solution, see also the equivalent formulation (6). It is well–known that z^* provides an upper bound to the optimal VaR value, see Rockafellar and Uryasev (2000). The computation time varied between 0.06 and 0.09 seconds across the different solvers.

Next, we performed comparative test runs with the problems corresponding to different sample sizes. Besides *CVaRMin*, we also solved the test problems with general–purpose LP solvers, where these solvers were used to solve the LP–equivalent (11). Additionally, we also solved the problems in their two–stage recourse form, by employing the solver *QDECOM*. The elapsed time values are summarized in Table 3.

In Table 3, *Cplex S* and *Cplex IP* stand for the dual simplex method and for the barrier method of *Cplex*, respectively. *OSL1 S* denotes the primal simplex method of *OSL*. *OSL* also has several built–in interior point methods. The corresponding entries are missing in Table 3, because the solver has crashed with the test problems with all the interior point options (error message: "more space is needed for the adjacency matrix").

Considering the above computing times, for the larger test problems *CVaRMin* clearly outperforms the other solvers by at least one order of magnitude. Second best was the interior–point solver *BPMPD*, with much better computing times than the rest of the solvers.

With *CVaRMin*, the number of cuts was between 25 and 49 for the test problems considered above.

For a proper interpretation of the results in Table 3, we have to discuss the selection of the solver parameters. By choosing them carefully for each test problem separately, it would most probably be possible to achieve significant improvements in the computing time ("tuning"). Let us remark that all general–purpose LP solvers have several parameters, typically including besides stopping tolerances also feasibility tolerances, see, for instance, Maros (2003). In fact, tuning means to look for appropriate parameter settings jointly for the solver parameters. Consequently, just prescribing a smaller stopping tolerance does not automatically lead to more precise results. For properly tuning solvers, detailed knowledge of the algorithm is needed in general.

Having in mind users who are not especially acquainted with the algorithms, the usual methodology in comparative testing of solvers is to perform the tests with a fixed choice of parameters. This is the case in our tests, too.

For the commercial GAMS solvers we have kept the default parameters. Our experimental solver *CVaRMin* has a single parameter, the stopping tolerance ε, see (18). This has been fixed as $\varepsilon = 10^{-10}$. *BPMPD* being a primal–dual interior point solver, the stopping rule includes testing the relative duality gap; therefore, we took for this parameter the same stopping tolerance $\varepsilon = 10^{-10}$ as for *CVaRMin*. For the other parameters of this solver, we kept the default values. Finally, the solver *QDECOM* has a single parameter, the cut tightness tolerance, for which we kept the default value 10^{-8}.

We now turn our attention to the objective values returned by the various solvers at termination, interpreted as optimal values. For the sake of simplicity, we will call them the optimal values returned by the solvers. From the purely mathematical point of view it is clear that, due to finite–precision arithmetic, these are, in general, merely approximations to the true optimal value.

Due to the fact that GAMS delivers results with at most 10 decimals, we compared the objective values rounded to this precision. A comparison with a higher relative precision can be found in the next section, where we discuss the results with randomly generated test problems.

Table 4. Test runs #1. Comparing optimal objective values, computed by *CVaRMin*, with those of the LP and NLP solvers

N	$\varepsilon = 10^{-8}$	$\varepsilon = 10^{-10}$	ΔLP	ΔNLP
500	0.0061163484	0.0061163484	0	0.037
1000	0.0062614463	0.0062614463	0	0.062
3000	0.0059029578	0.0059029616	0	0.001
5000	0.0059944576	0.0059944594	0	0.017
10000	0.0059716694	0.0059716704	10^{-8}	0.013
20000	0.0058584314	0.0058584365	0	0.006

The general–purpose LP solvers *BPMPD*, *Cplex S*, *Cplex IP*, and *OSL1 S* delivered the same objective value for each of the test problems, with a sole exception: the objective value for *BPMPD* and for sample size 5'000 differs by $4 \cdot 10^{-10}$ from the optimal objective value obtained from the rest of the LP solvers. Considering *CVaRMin*, the optimal objective values are the same as those of the LP solvers, with one exception: for the test problem with $N = 10'000$ the difference is 10^{-10}. Finally, for *QDECOM* the objective values were the same as those of the LP solvers for sample sizes 500 and 1'000; for the larger sample sizes the deviation varied between 10^{-8} and 10^{-7}.

Table 4 shows the optimal objective values obtained via *CVaRMin*. Besides the stopping tolerance $\varepsilon = 10^{-10}$, which was used throughout in the computations, we have also included results with a higher stopping tolerance $\varepsilon = 10^{-8}$ for the purpose of giving an impression on the influence of the stopping tolerance on the solution. The fourth column (ΔLP) in the table displays the relative deviation of the values in the third column ($\varepsilon = 10^{-10}$), with respect to the objective value delivered by the majority of the LP solvers. The fifth column (ΔNLP) shows the relative deviation with respect to the objective value from the NLP formulation. For the sake of completeness: the relative deviation of an approximate value f_{approx} with respect to a base value $f^* \neq 0$ is computed throughout according to $\dfrac{|f_{\text{approx}} - f^*|}{|f^*|}$.

The optimal objective values of the six test–problems are displayed in Figure 2. The points corresponding to the five values have been connected by straight line segments; the horizontal dotted line corresponds to the optimal objective value 0.0058959347 of the NLP problem.

Test runs #2. Portfolio optimization: several generated samples
In the runs so far, for each one of the selected sample sizes we have generated a single sample and used this for setting up a corresponding test problem. The question arises, whether the favorable running times for *CVaRMin* are due to blind chance, by having obtained test problems, which are especially favorable for the algorithm.

For testing this, we generated a test problem battery as follows: for each of the sample sizes 500, 5'000, 10'000, and 20'000, respectively, we generated 10 different samples by choosing different seeds for the random number generator

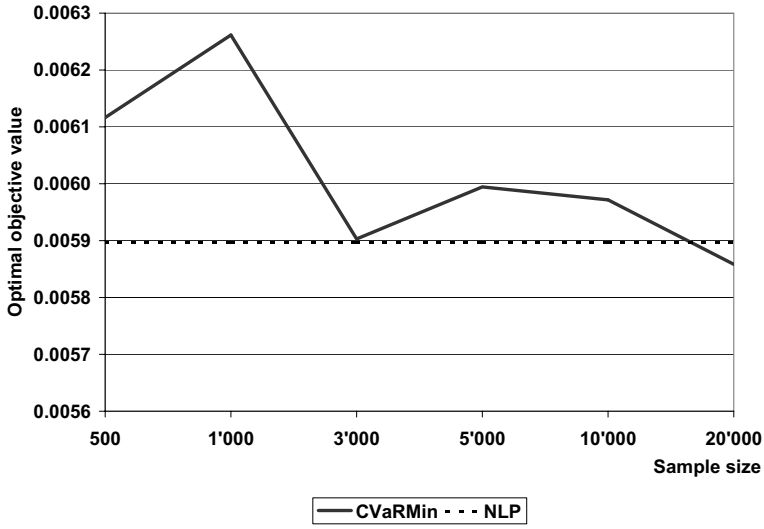

Fig. 2. Optimal objective values corresponding to different sample sizes

Table 5. Test runs #2. Basic statistics for results from runs with different samples (varying seeds for the generator), computed by the solver *CVaRMin*; the optimal NLP value is 0.005896

	N=500	N=5'000	N=10'000	N=20'000
min (sec)	0.009	0.032	0.067	0.122
max (sec)	0.043	0.051	0.105	0.191
mean (sec)	0.015	0.045	0.088	0.162
stdev (sec)	0.010	0.007	0.013	0.027
min (ov)	0.004512	0.005546	0.005673	0.005739
max (ov)	0.006246	0.006557	0.006135	0.006059
mean (ov)	0.005230	0.005978	0.005859	0.005893
stdev (ov)	0.000591	0.000344	0.000133	0.000124
mrd (ov)	23.5%	11.2%	4.1%	2.8%

(altogether 40 different seeds). The sample generation method was the same as for the base case. The samples generated this way have then been utilized for setting up the test problems for the portfolio optimization problem. Consequently, this test problem battery consists of 40 test problems.

We let *CVaRMin* run on these test problems; a basic statistics concerning elapsed time **(sec)** is given in Table 5, where **stdev** abbreviates standard deviation. The results clearly indicate that the computing times for *CVaRMin*, used for comparison in Table 3, are most likely typical for the solver.

As a byproduct of these runs, we can also get an impression concerning the impact of varying samples on the optimal objective value. Table 5 also shows a basic statistics concerning the optimal objective values (**ov**) obtained in the runs.

The last row in this table displays the maximal relative deviations (**mrd**) between the objective values corresponding to the sampled problems and the optimal objective value of the exact NLP problem. The maximal relative deviations for the sample sizes $N = 500$ and $N = 5'000$ are quite high; but the deviations become smaller with increasing sample sizes as intuitively expected.

This indicates the following: when solving (19) with a continuous distribution via sampling and by solving the sampled problem, one must be cautious; just solving one such approximating problem is quite dangerous. Either statistical analysis is needed, or at least a large sample size should be chosen. In any case, a fast solver can be well–utilized for carrying out the necessary computations.

Test runs #3. Portfolio optimization: parametric results
In these tests, we have carried out computations with varying the key problem parameter α. The results from such computations are of primary interest in financial portfolio optimization. From the algorithmic point of view, this means solving a sequence of CVaR optimization problems, with varying values of a problem parameter.

We considered the portfolio optimization problem (19) for different values of the probability level α. For this reason, we took an equidistant subdivision of the interval [0.6, 0.999] into 99 sub–intervals, and solved the sampled version of (19) for the 100 different values of $\alpha \in [0.6, 0.999]$. The sample size was $N = 20'000$, and we utilized our solver *CVaRMin* for the computations. Figure 3 shows the CVaR values corresponding to the optimal solution, in dependency on the probability level α.

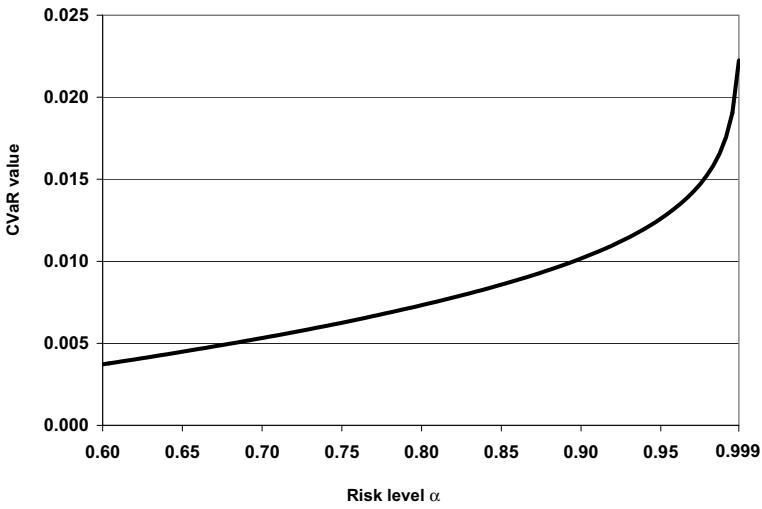

Fig. 3. Optimal CVaR value versus α, $N = 20'000$

When solving a sequence of closely related optimization problems, the computations can usually be speeded up, by utilizing results from the previous run for providing appropriate start–data for the current run ("warm start"). In our case, we tested two different strategies. First, we have carried out the computations according to the basic description of the method in Section 3, that is, in the initialization *Step 0* we started with a single cut, corresponding to the expected value ("cold start"). In a second version, for $\nu > 1$, we modified the initialization in *Step 0* as follows. As a starting set of cuts, we kept all cuts from the previous optimization, for which the optimal Lagrange multiplier in the relaxed master problem (13) was non–zero ("warm start"). This set of cuts is clearly a subset of all active cuts at the optimal solution of (13). With a cold start, the overall computing time for the 100 problems was 20.48 sec, whereas with the above described warm start the computing time was reduced to 16.32 sec, a saving of $\sim 20\%$ in elapsed time. Of course, this has to be tested on a large number of test problems, before definitive conclusions can be drawn concerning the efficiency of the warm start described above. Although not directly related to the performance of *CVaRMin*, for readers interested in financial portfolio optimization, we also present Figure 4, which displays the dependency of the asset weights in the portfolio on α. For all α–values considered, wealth is diversified over the three asset classes *Swiss equities*, *Swiss franc bonds*, and *global bonds*, that is, the asset classes *European equities* and *world equities* have zero weight and therefore they do not appear in Figure 4. On the one hand, this shows that CVaR minimization resulted in a diversified portfolio. On the other hand, the results are in accordance with financial intuition, the asset classes with zero weight are the most risky among the asset classes considered here.

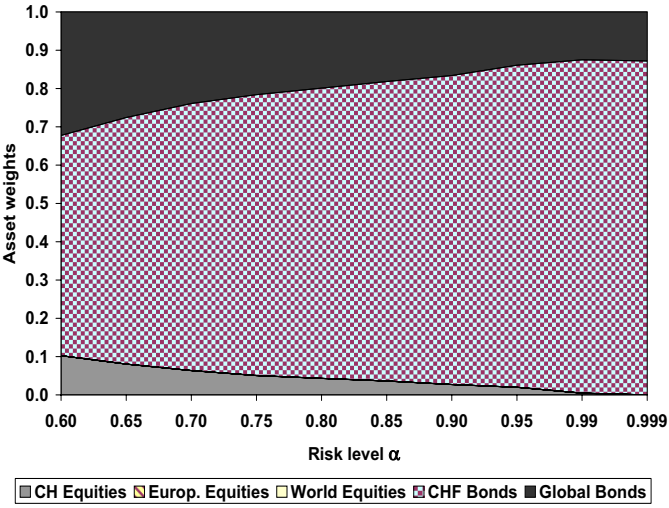

Fig. 4. Asset weights in the portfolio versus α

Note that the general formulation of (19) in finance also involves a weighting factor for the risk term in the objective function. This is interpreted as a risk–aversion parameter; varying this parameter leads to the risk–return frontier. The parametric computations were performed in the same manner as for varying α and it turned out that the same three asset classes appear with positive weights throughout.

4.2 Randomly generated test problems

The computational results reported above are all with respect to the portfolio optimization problem (19). The special structure of this problem and our assumption concerning normal distribution may imply that our test problems are biased in favor of our algorithm. For further testing the performance of *CVaRMin*, we took randomly generated test problems.

The sole purpose of this section is to illustrate the performance of *CVaRMin* on larger test problems as in the tests so far. We generated instances of the prototype problem (5), with $\mathcal{P} = \{ x \mid Ax = b,\ x \geq 0 \}$, where A is an $(m \times n)$ matrix and b, c have compatible dimensions. In the test problems, we chose $m = 10$ and $n = 50$. Thus, the random variable η is 50–dimensional; ξ was chosen as constant $\xi \equiv 0$.

The elements of the matrix A were randomly generated, according to the uniform distribution over the range $[-2, 2]$; the density of non–zeros is 50%. The right–hand–side vector b has been chosen such that $\mathcal{P} \neq \emptyset$ holds and, finally, the choice of the objective vector c ensures the existence of an optimal solution.

For the probability distribution of η, we took a multivariate shifted lognormal distribution. First, the parameters of the underlying multivariate normal distribution were computed. The expected values were generated according to a uniform distribution over $[-2, 2]$. The standard deviations were computed by multiplying the expected values with some factors. These factors were generated according to the uniform distribution over $[1.2, 1.6]$. The correlation matrices were generated by employing the method of Marsaglia and Olkin (1984). For the reliability level $\alpha = 0.9$ was chosen.

The parameters of 10 different CVaR minimization problems were computed according to the above scheme first.

The test problem batteries themselves resulted via sampling from the lognormal distribution. The random variates y for the multivariate normal distribution were generated by employing the same sampling method, based on the factorization of the covariance matrix, as described in Section 4.1. These were subsequently transformed component–wise according to $\hat{y}_i := e^{y_i}$ and finally shifted to the left by e_i^{μ}, with μ being the expected value of the underlying normal distribution. The sole purpose of this shift was to achieve both positive and negative realization values. Considering the goal of this test, namely just testing solver efficiency, we did not care for a statistically meaningful shift.

We generated two test problem batteries, each consisting of 10 test problems, corresponding to the sample sizes $N = 10'000$ and $N = 20'000$, respectively.

Table 6. Test runs #4. Elapsed time summary (sec) for $N = 10'000$

	BPMPD	Cplex S	Cplex IP	CVaRMin	OSL1 S
T1a	6.22	21.50	15.59	0.70	409.14
T2a	6.80	20.89	21.89	0.46	376.23
T3a	8.89	24.77	24.20	1.16	374.35
T4a	8.03	19.98	13.51	0.55	360.50
T5a	7.11	17.52	17.91	0.26	384.24
T6a	6.17	19.33	14.83	0.59	372.61
T7a	6.52	22.55	13.47	0.50	353.61
T8a	16.52	22.59	16.16	0.48	345.34
T9a	37.25	20.64	18.55	0.46	452.37
T10a	8.94	20.20	15.16	0.37	311.88

Test runs #4. Randomly generated test problems
We ran the solvers with the same parameter settings as before, in particular, we employed the same stopping tolerances. The computing times are displayed in Tables 6 and 7 and show the clear superiority of the solver *CVaRMin*. The number of cuts (iterations) varied between 24 and 106.

Table 7. Test runs #4. Elapsed time summary (sec) for $N = 20'000$

	BPMPD	Cplex S	Cplex IP	CVaRMin	OSL1 S
T1b	14.45	84.09	89.61	1.33	1826.58
T2b	13.97	82.27	97.92	0.95	1803.64
T3b	19.59	96.31	65.20	2.26	1673.38
T4b	23.92	82.16	64.19	1.11	1782.70
T5b	21.33	71.94	99.28	0.59	1749.61
T6b	14.44	75.08	59.38	1.08	1691.39
T7b	127.84	85.25	54.61	0.86	1694.31
T8b	15.08	88.17	96.30	0.74	1680.39
T9b	160.14	81.23	59.78	0.54	2174.45
T10b	25.12	82.70	51.95	0.96	1418.35

Notice that the tables do not contain results with *QDECOM*. The reason is this. For the battery with $N = 10'000$, the solver could just solve 4 problems out of 10, for the rest it seemed to be cycling and the run was terminated at a time limit of 10 minutes.

The question arises, whether the excellent elapsed times for *CVaRMin* are perhaps due to imprecise results, in comparison with the general–purpose LP solvers. We, therefore, chose one of the LP solvers as a basis; we decided to choose a commercial solver for this purpose and took *OSL1 S*. Table 8 shows the results. The second and third columns display the objective values from *OSL1 S*, rounded to 10 decimals, and corresponding to $N = 10'000$ and $N = 20'000$, respectively. The other columns show the absolute deviation for the objective values. The columns labelled as δBP, δCP, and δCV show the absolute deviation in objective value, with respect to OSL/20'000, for *BPMPD*, *CPlex* and *CVaRMin*, respectively. The two

Cplex–variants returned the same values, most probably due to a crossover to the simplex method after the barrier iterations. Thus, the heading for this solvers is just CP. The last two rows contain the maximal absolute deviation **mad** and the maximal relative deviation **mrd**, respectively. Finally, we employed the notation kEd for $k \cdot 10^d$, which is quite common in numerical analysis.

Notice that the difference in objective values is in general not zero for different solvers, even if comparing two commercial LP solvers like *Cplex* and *OSL1*. The deviations of our solver *CVaRMin* are in the same range, as the deviations for the general LP solvers.

Let us point out that the deviations in the table can by no means be interpreted as errors, by mistaking the objective values delivered by *OSL1 S* as the true values (with zero error), rounded to 10 decimal places. Due to finite precision arithmetic, none of the LP solvers is capable to compute the mathematically exact solution for all LP's, in general (see also the discussion regarding solver parameters and various tolerances above).

Table 8. Test runs #4. Objective values at termination

	OSL/10'000	OSL/20'000	δBP	δCP	δCV
T1	2968.0109913872	2975.1675789166	4E-06	4E-06	4E-06
T2	3146.0517148887	3148.2881231466	2E-05	3E-10	1E-08
T3	2969.4916454730	2957.0655695859	2E-05	6E-07	6E-07
T4	2738.8821014082	2710.4796681820	4E-05	8E-09	7E-08
T5	3157.1759883929	3128.5835564060	9E-06	1E-08	9E-08
T6	2979.6663741827	2977.2050961986	1E-05	3E-08	3E-08
T7	3384.0469230574	3394.8485933748	4E-06	2E-08	1E-08
T8	4024.0331972310	4023.5791142508	8E-06	9E-09	6E-08
T9	4649.6235981131	4640.9993879112	7E-07	3E-08	5E-08
T10	1929.0589768021	1929.9903118863	5E-06	5E-08	3E-08
mad			4E-05	4E-06	4E-06
mrd			2E-08	1E-09	1E-09

5 Conclusions

In this paper, we considered one–stage optimization problems involving the minimization of conditional value–at–risk (CVaR) in the objective. We proposed to solve these problems via reformulating them as two–stage stochastic optimization problems with recourse and presented an algorithm. The method was derived by adapting the L–shaped method for two–stage recourse problems, to the special structure of CVaR minimization problems.

We have implemented our solution approach as the solver *CVaRMin*. In the paper we presented comparative computational results with several test problems.

First we considered a portfolio optimization problem of the CVaR minimization type, with 5 random variables and assumed a multivariate normal distribution. Due to our assumptions, an equivalent NLP formulation exists, which can also be solved by general–purpose NLP solvers, thus providing an optimal value for comparison purposes.

We generated several test problems by sampling and compared the performance of our solver with general–purpose LP solvers. In our experiments, *CVaRMin* outperformed all LP solvers involved in the comparison; for the largest test–problem the elapsed time for *CVaRMin* was at least by one order of magnitude better than the elapsed time for the rest. The solution time for *CVaRMin* had the same order of magnitude as the solution times for the NLP problem formulation.

For testing the effect of sampling on the solution times of *CVaRMin*, we generated portfolio optimization test problems by sampling with 40 different seeds. The results indicate that the performance of *CVaRMin* does not significantly depend on choosing different samples. Concerning optimal values across different samples, the obtained results suggest that, even for 5 random variables, a sample size of $N = 5'000$ may lead to quite inaccurate results. Thus it is likely that for larger numbers of random variables quite high sample sizes are needed for obtaining acceptable results.

We also presented computational results for varying α and proposed a "warm start" strategy for solving a sequence of CVaR minimization problems with varying parameter. According to this test, the strategy appears to lead to a significant reduction in computing time.

Finally, we presented our computational results with 20 randomly generated test problems involving 50 random variables and a shifted multivariate lognormal distribution. In these tests, *CVaRMin* outperformed again all general purpose LP solvers by at least one order of magnitude in the solution time. The deviations in optimal objective values were in the same range, as the deviations across the different LP solvers.

The computational results indicate that our algorithm is quite promising. These results are, however, by no means conclusive in this respect; for this much more extensive tests would be needed.

It is clear that for CVaR minimization problems with random–variable dimensions employed in our computations, the approach via solving the LP–equivalent is in many practical cases quite satisfactory from the point of view of computing time. Under the assumption of a multivariate normal distribution, solving the NLP formulation via general–purpose NLP solvers may be a competitive alternative. This was so in our tests with 5 random variables, where the NLP solvers outperformed all LP solvers, for the larger sample sizes. Let us shortly discuss application areas where a fast solution method can be of direct practical interest.

In portfolio optimization, one such area could be the optimization of large hedge portfolios with possibly thousands of assets (see, for instance, Rockafellar and Uryasev (2000)). Another important application area, where a fast solver could

help, is parametric computation related to CVaR minimization (in financial terms this means computing "frontiers"), especially for large numbers of random variables and large sample sizes.

CVaR minimization problems can also be used as building blocks for algorithms aiming at minimizing VaR, see Larsen, Mausser, and Uryasev (2002). Thus, an efficient method for solving the CVaR subproblems may lead to significantly increased performance of the overall method. For a recent different sequential approach for minimizing VaR see Pang and Leyffer (2004). Besides finance, minimizing VaR and CVaR gets increasing attention also in other application areas, see Chen, Daskin, Shen, and Uryasev (2005). This may lead to large–scale CVaR minimization problems for which efficient special–purpose solvers might be needed.

Our future research plans include testing the algorithm with financial optimization problems involving larger amounts of assets and realizations, extending the scope of *CVaRMin* for optimization under CVaR constraints, refining the warm start strategy for computing frontiers, and developing extensions to mixed–integer problems.

Acknowledgements. The authors gratefully acknowledge the helpful comments and suggestions of the anonymous referees.

References

Acerbi C (2002) Spectral measures of risk: a coherent representation of subjective risk aversion. Journal of Banking & Finance 26:1505–1518.

Acerbi C, Tasche D (2002) On the coherence of expected shortfall. Journal of Banking & Finance 26: 1487–1503.

Artzner P, Delbaen F, Eber JM, Heath D (1999) Coherent measures of risk. Mathematical Finance 9: 203–228.

Benders JF (1962) Partitioning procedures for solving mixed-variables programming problems. Numer. Math. 4:238–252, 1962. Re–published in this Journal, Vol. 2, Number 1, 2005.

Birge JR, Louveaux F (1997) Introduction to Stochastic Programming. Springer Series in Operations Research. Springer-Verlag, Berlin/Heidelberg.

Brooke A, Kendrick D, Meeraus A, Raman R (1998) GAMS. A user's guide. Technical report, GAMS Development Corporation, Washington DC, USA. It can be downloaded from http://www.gams.com/docs.

Chen G, Daskin MS, Shen ZM, Uryasev S (2005) A new model for strategic facilities location. Naval Research Logistics: Special Issue on Applications of Financial Engineering in Operations, Production, Services, Logistics, and Management. Under review.

Elton EJ, Gruber MJ, Brown SJ, Goetzmann WN (2003) Modern portfolio theory and investment analysis. John Wiley & Sons, sixth edition.

Gams Development Corporation (2004) GAMS. The solver manuals. Washington, DC.

Jorion PH (1996) Value at Risk: a new benchmark for mesuring derivatives risk. Irwin Professional Publications.

Kall P, Mayer J (1996) SLP-IOR: An interactive model management system for stochastic linear programs. Math. Prog. B 75:221–240.

Kall P, Mayer J (2004b) SLP-IOR User's Guide. University of Zurich. Version 2.2.1.

Kall P, Mayer J (2005a) Some insights into the solution algorithms for SLP problems. Ann. Oper. Res. (to appear).

Kall P, Mayer J (2005b) Stochastic linear programming. Models, theory and computation. Springer-Verlag, New York, Berlin, Heidelberg.

Kall P, Mayer J (2005c) Building and solving stochastic linear programming models with SLP-IOR. In: Wallace SW, Ziemba WT (eds) Applications of Stochastic Programming, Series in Optimization, chapter 6, pp. 79–93. MPS SIAM, SIAM, Philadelphia.

Kall P, Wallace SW (1994) Stochastic programming. John Wiley & Sons, Chichester.

Klein Haneveld WK, Van der Vlerk MH (2002) Integrated chance constraints: reduced forms and an algorithm. SOM Research Report 02A33, University of Groningen. To appear in this Journal.

Krokhmal P, Palmquist J, Uryasev SP (2002) Portfolio optimization with conditional value-at-risk objective and constraints. The Journal of Risk 4.

Larsen N, Mausser H, Uryasev S (2002) Algorithms for optimization of value–at–risk. In: Pardalos P, Tsitsiringos VK (eds) Financial Engineering, E–commerce and Supply Chain, pp. 19–46. Kluwer Academic. Publishers.

Linderoth J, Shapiro A, Wright S (2005) The empirical behavior of sampling methods for stochastic programming. Ann. Oper. Res. (to appear).

Markowitz H (1959) *Portfolio selection. Efficient diversification of investments*. John Wiley & Sons

Maros I (2003) *Computational Techniques of the Simplex Method*. Kluwer Academic Publishers

Marsaglia G, Olkin I (1984) Generating correlation matrices. SIAM J. on Scientific and Statistical Computations 5:470–475.

Mayer J (1998) Stochastic Linear Programming Algorithms: A Comparison Based on a Model Management System. Gordon and Breach Science Publishers.

Mészáros Cs (1997) The augmented system variant of IPMs in two–stage stochastic linear programming computation. Eur. J. Oper. Res. 101:317–327.

Murtagh BA, Saunders MA (1978) Large scale linearly constrained optimization. Math. Prog. 14: 41–72.

Pang JS, Leyffer S (2004) On the global minimization of value–at–risk. Opt. Methods and Software 19:611–631.

Pflug GCh (2000) Some remarks on the Value–at–Risk and the Conditional Value–at–Risk. In: Uryasev SP (ed) Probabilistic constrained optimization, methodology and applications, pp. 272–281. Kluwer Academic Publishers.

Ripley BD (1987) Stochastic simulation. John Wiley & Sons, New York.

Rockafellar TR, Uryasev SP (2000) Optimization of Conditional Value–at–Risk. Journal of Risk 2: 21–41.

Rockafellar TR, Uryasev SP (2002) Conditional Value–at–Risk for general loss distributions. Journal of Banking & Finance 26:1443–1471.

Ruszczyński A (1986) A regularized decomposition method for minimizing a sum of polyhedral functions. Math. Prog. 35:309–333.

Ruszczyński A, Świętanowski A (1997) Accelerating the regularized decomposition method for two stage stochastic linear programming problems. Eur. J. Oper. Res. 101:328–342.

Van Slyke R, Wets RJ-B (1969) *L*-shaped linear programs with applications to optimal control and stochastic linear programs. SIAM J. Appl. Math. 17:638–663.

Zivot E, Wang J (2003) *Modeling financial time series with S-PLUS*. Springer-Verlag, New York, Berlin, Heidelberg.