Beliakov, Gleb and Bagirov, Adil 2006, Non-smooth optimization methods for computation of the conditional value-at-risk and portfolio optimization, *Optimization*, vol. 55, no. 5&6, pp. 459-479.

Optimization
Vol. 00, No. 00, Month-Month 200x, 1–25

Nonsmooth optimization methods for computation of the Conditional Value-at-Risk and portfolio optimization

Gleb Beliakov, School of Information Technology, Deakin University, 221 Burwood Hwy, Burwood 3125, Australia, gleb@deakin.edu.au
Adil Bagirov, Centre for Informatics and Applied Optimisation, School of Information Technology and Mathematical Sciences, University of Ballarat, Ballarat 3353, Australia, a.bagirov@ballarat.edu.au
(v4.0 released January 2005)

We examine numerical performance of various methods of calculation of the Conditional Value at Risk (CVaR), and portfolio optimization with respect to this risk measure. We concentrate on the method proposed by Rockafellar and Uryasev in (Journal of Risk, 2 (2000), 21-41), which converts this problem to that of convex optimization. We compare the use of linear programming techniques against a non-smooth optimization method of the discrete gradient, and establish the supremacy of the latter. We show that non-smooth optimization can be used efficiently for large portfolio optimization, and also examine parallel execution of this method on computer clusters.

Key words: Non-smooth optimization, Conditional Value at Risk, Expected shortfall, portfolio optimization, numerical optimization.

AMS 2000 Mathematics Subject Classification: 90C05, 90C30, 90C56, 91B28

1 Introduction

Portfolio optimization facilitates the process of fund allocation such that the optimal weightings in competing securities may be found whilst satisfying choice constraints of risk and return. Value-at-risk (VaR) has become a standard measure used in financial risk management due to its conceptual simplicity, computational facility, and ready applicability. However, many authors claim that VaR has several conceptual problems, notably it is considered as an inconsistent risk measure [1].

To alleviate the problems inherent in VaR, Artzner et al. [2] have proposed the use of expected shortfall. Expected shortfall, or conditional VaR (CVaR), is defined as the conditional expectation of loss given that the loss is beyond the VaR level. Expected shortfall is proved to be sub-additive, which assures its coherence as a risk measure [1–3]. On these grounds, some practitioners have been turning their attention toward expected shortfall and away from

VaR.

Recently Rockafellar and Uryasev [4, 5] have presented a new technique, which allows one to compute both measures of risk (VaR and CVaR) simultaneously by using linear programming and non-smooth optimization techniques. Furthermore, they proved that portfolio optimization can be performed simultaneously with calculating the optimal VaR and CVaR. Their method relies on minimizing a certain auxiliary objective function F_β, defined in the sequel, which is non-smooth, but convex for linear portfolios. In the latter case, the optimization problem can be reduced to a large scale linear programming problem.

Our main goal is to show that non-smooth optimization methods can be used very effectively for portfolio optimization. We examine the method of Rockafellar and Uryasev from the numerical point of view. We compare various optimization techniques that can be used to compute VaR and CVaR, as well as determine the optimal portfolio. In particular we will show that using the discrete gradient method of non-smooth optimization (DG) [6] provides significant computational advantages over linear programming methods used in [4], including scalability for large portfolios, parallel computations, and the ability to treat non-linear portfolios. We do not make any assumptions about the normality of the distribution of portfolio losses.

The next section provides the basic definitions and outlines the method of Rockafellar and Uryasev [4]. In section 3 we discuss two approaches to numerical solution of the portfolio optimization problem: by using non-smooth optimization and by converting it to a large scale linear programming problem. We describe the non-smooth optimization approach and the discrete gradient method in sections 4 and 5. The performance of numerical algorithms is analyzed and compared in section 6, by using a number of test cases. In section 7 we discuss the use of computer clusters for portfolio optimization problem and our parallelization strategy. We present conclusions at the end of the paper.

2 Preliminaries

Consider a portfolio of market instruments of size n, represented by the relative weightings of these instruments $w \in \mathbb{R}^n, w_i \geq 0, \sum w_i = 1$. Let vector $x \in \mathbb{R}^n$ denote future returns of these instruments, which are random variables with some underlying distribution with density $p(x)$. The loss of the portfolio $L(w, x)$ (a negative return) is also a random variable which depends on the market uncertainties x and weightings w. For example, for a linear portfolio $L(w, x) = -w^t x$.

The Value-at-Risk (VaR), or more precisely β-VaR, is defined as the maximum loss $L(w, x)$ such that with probability β (typically 95%) portfolio loss

will not exceed α. The probability of $L(w,x)$ not exceeding α is given as

$$\beta = P(L(w,x) \leq \alpha) = P(\alpha) = \int_{L(w,x)\leq\alpha} p(x)\ dx, \tag{1}$$

VaR is expressed as the β-th quantile of P

$$\alpha_\beta = \min\{\alpha \in \mathbb{R} : P(\alpha) \geq \beta\}. \tag{2}$$

CVaR or β-CVaR, also referred to as the mean excess loss, mean shortfall, or tail VaR, is defined (for continuous distributions) as the conditional expected loss subject to the condition that it falls below β-VaR. β-CVaR is defined as the expectation

$$\begin{aligned}\psi_\beta &= E(L(w,x)|L(w,x) \geq \alpha_\beta) \\ &= (1-\beta)^{-1} \int_{L(w,s)\geq\alpha_\beta} L(w,x)p(x)\ dx \\ &= (1-\beta)^{-1} \int_{\mathbb{R}^n} (L(w,x)-\alpha_\beta)_+ p(x)\ dx,\end{aligned} \tag{3}$$

where $t_+ = \max\{t, 0\}$.

CVaR possesses a number of desired mathematical properties, notably convexity and subadditivity, and hence is a more attractive measure of risk than VaR [3, 4, 7, 8] (VaR is subadditive only for Gaussian distributions and some other special cases). Unlike VaR, which may discourage diversification as it underestimates the risks associated with the individual securities [3], CVaR correctly embodies the reduction of risks associated with diversification.

For discontinuous distributions, there are several equivalent definitions of CVaR [5], notably it can be defined as a weighted average of VaR and conditional expected losses exceeding VaR, CVaR+, see [5]. In this general case CVaR is still a coherent measure of risk.

We consider the following portfolio optimization problem: find the optimal weightings w, such that the risk associated with the portfolio is minimized. The feasible domain is the unit simplex

$$S = \{w \in \mathbb{R}^n : w_i \geq 0, \sum w_i = 1\},$$

however it is common to restrict it further, for instance by considering only portfolios with the expected return no less than some value R, i.e., $L(w, E(x)) \leq -R$.

There are many measures of risk that can be used for portfolio optimization [7, 9], VaR and CVaR are among them. In this study, following a number of authors [3–5, 7], we will optimize the portfolio with respect to CVaR.

Rockafellar and Uryasev [4, 5] have presented a new technique for efficient computation of both VaR and CVaR, as well as for porftolio optimization, by using non-smooth optimization. Their method represents α_β and ψ_β in terms of a continuous auxiliary function F_β, and finds the respective risk measures by minimizing F_β. F_β has been given as

$$F_\beta(\alpha) = \alpha + (1-\beta)^{-1} \int_{x \in \mathbb{R}^n} (L(w,x) - \alpha)_+ p(x)\ dx, \tag{4}$$

where we drop the index β in α_β to simplify the notation. Theorem 1 in [4](p. 24-25) establishes β-CVaR and β-VaR, with respect to F_β, as

$$\psi_\beta = \min F_\beta(\alpha), \tag{5}$$

and

$$\alpha_\beta = \text{lower bound of} \arg\min F_\beta(\alpha). \tag{6}$$

The convenience of the formulae above is that both measures of risk are calculated simultaneously. Furthermore, Theorem 2 in [4] (p. 25-26) establishes that minimizing β-CVaR associated with w over all $w \in S$ is equivalent to minimizing $F_\beta(w, \alpha)$ over all $(w, \alpha) \in S \times \mathbb{R}$:

$$\min_w \ \psi_\beta(w) = \min_{w,\alpha} \ F_\beta(w, \alpha), \tag{7}$$

Thus calculation of both measures of risk, and portfolio optimization with respect to CVaR can be performed simultaneously. We note that the above equations have been derived irrespective of the distribution $p(x)$, which is not necessarily normal or log-normal.

The main attractiveness of using (7) is that F_β is a convex continuously differentiable function of α, and for convex loss functions $L(w, x)$ (e.g., linear portfolios), it is also convex w.r.t. w. Therefore its minimum can be found by methods of convex optimization. This is in sharp contrast with optimizing portfolio w.r.t. VaR, in which case the objective function is non-convex, and involves many local minima.

3 Calculation methods

The evaluation of the objective function F_β involves a multidimensional integral over $\mathbb{R}^n$, which can be calculated by using Monte-Carlo method. For this we sample the probability distribution of x according to its density $p(x)$. The sampling generates a collection of q vectors $x_1, \ldots, x_q \in \mathbb{R}^n$, called scenarios. F_β is approximated as

$$\hat{F}_\beta(w, \alpha) = \alpha + \frac{1}{q(1-\beta)} \sum_{i=1}^{q} (t_i - \alpha)_+, \tag{8}$$

where $t_i = L(w, x_i)$ is the loss of the portfolio under the scenario i. $\hat{F}_\beta$ is convex and piecewise linear with respect to α. It is also convex with respect to w, as long as $L(w, x)$ is convex. For a linear portfolio, $\hat{F}_\beta$ is convex and piecewise linear with respect to w.

We consider two problems.

Problem A Calculation of CVaR and VaR for a fixed portfolio.

$$\begin{aligned} &\min_\alpha \hat{F}_\beta(w, \alpha), \\ &\text{s.t.} \quad w \text{ fixed.} \end{aligned} \tag{9}$$

Problem B Calculation of the optimal portfolio.

$$\begin{aligned} &\min_{\alpha, w} \hat{F}_\beta(w, \alpha), \\ &\text{s.t.} \quad w \in S \cap D, \end{aligned} \tag{10}$$

where the feasible domain is the intersection of the unit simplex S and a compact subset D, representing user specified requirements, such as having a portfolio with the mean return at least R, i.e.,

$$D = \{w \in S : L(w, E(x)) \leq -R\}.$$

As a special case, we consider a linear portfolio, with $L(w, x) = -w^t x$. In this case both problems A and B can be reformulated as linear programming problems [4].

Problem A' Calculation of CVaR and VaR for a fixed portfolio.

$$\begin{gathered} \min_{\alpha,\varepsilon} \alpha + \frac{1}{q(1-\beta)} \sum_{i=1}^{q} \varepsilon_i \\ \text{s.t. } \alpha + \varepsilon_i \geq t_i = L(w, x_i),\ i = 1, \ldots, q, \\ \varepsilon_i \geq 0, \alpha \text{ urs}, \\ w \text{ fixed}. \end{gathered} \tag{11}$$

Problem B' Calculation of the optimal linear portfolio.

$$\begin{gathered} \min_{\alpha,w,\varepsilon} \alpha + \frac{1}{q(1-\beta)} \sum_{i=1}^{q} \varepsilon_i \\ \text{s.t. } \alpha + \varepsilon_i + \sum_{j=1}^{n} w_j x_{ij} \geq 0,\ i = 1, \ldots, q, \\ w \in S \cap D \\ \varepsilon_i \geq 0, \alpha \text{ urs}. \end{gathered} \tag{12}$$

Here x_{ij} denotes the j-th component of the vector x_i.

Thus for each problem A and B we have two approaches: (i) to solve problem A (or B) directly using non-smooth optimization, and (ii) to solve problem A' (or B') using linear programming methods. In the latter case we can use the simplex method or the interior point method. Note that in Problem A' the system of constraints is sparse, while in Problem B' it is dense.

For an accurate approximation of F_β we need to use a large number of scenarios, especially for large portfolios. Therefore problems A' and B' will have a very large number of variables, namely $q+1$ and $n+q+1$. In contrast, Problems A and B have one and $n+1$ variables. It is not obvious which approach will be more efficient computationally, and one of the purposes of this study is to provide a comparative analysis of these techniques. Before we do this, however, we will describe the non-smooth optimization technique used in our study.

4 Non-smooth optimization approach

Portfolio optimization Problem B is a non-smooth optimization problem with linear equality and inequality constraints. A comprehensive description of non-smooth analysis can be found, for example, in [10]. There are different generalizations of the gradient that were proposed and studied by many authors. Two such generalizations: the Clarke subdifferential and Demyanov-Rubinov quasidifferential are widely used. In this study we employ the discrete gradient method, which uses a finite difference approximation to Clarke subdifferential.

The detailed description of the discrete gradient method as a tool for unconstrained non-smooth minimization was given in [6, 11–13]. Since the portfolio optimization problem involves a set of linear constraints, we first outline how such a problem is reduced to a non-smooth unconstrained optimization problem, to which we will subsequently apply the discrete gradient method.

We consider the following minimization problem with linear equality

$$\text{minimize} \quad f(x) \tag{13}$$

$$\text{subject to } x \in X = \{y \in \mathbb{R}^n : \ Ay = b\} \tag{14}$$

where the objective function f is assumed to be convex and in general, nonsmooth, A is a $m \times n$ matrix, $b \in \mathbb{R}^m$. Without loss of generality we assume that the rank of the matrix A is equal to m and $m < n$.

In this case one can solve the system of linear equations (14). Since $m < n$ then we can divide variables $x_1, \ldots, x_n$ into two parts: $x = (x_B, x_N)$ where $x_B \in \mathbb{R}^{n-m}$, $x_N \in \mathbb{R}^m$. Then one can present the matrix as follows:

$$A = (A_1, A_2)$$

where A_1 is a $m \times (n-m)$ matrix consisting of columns of the matrix A corresponding to variables x_B and A_2 is a $m \times m$ matrix consisting of columns of the matrix A corresponding to variables x_N and A_2 is not singular. Then the system (14) can be rewritten as:

$$A_1 x_B + A_2 x_N = b.$$

One can solve this system of linear equations with respect to non-basic variables x_N:

$$x_N = A_2^{-1}(b - A_1 x_B).$$

Thus we can represent the non-basic variables x_N as follows:

$$x_N = B x_B + b^1,$$

where

$$b^1 = A_2^{-1} b, \quad B = -A_2^{-1} A_1.$$

The objective function f in problem (13) can be rewritten as

$$f(x) = f(x_B, x_N) = f(x_B, Bx_B + b^1).$$

We define the following function

$$h(y) = f(y, By + b^1), \quad y \in \mathbb{R}^{n-m}.$$

PROPOSITION 4.1 *Let f be a convex proper function on $\mathbb{R}^n$. Then the function h is a proper convex on $\mathbb{R}^{n-m}$.*

Proof This result is known in convex analysis in more general cases, however we shall provide a short direct proof here.

$$\begin{aligned} h(\alpha y + (1-\alpha)z) &= f(\alpha y + (1-\alpha)z, B(\alpha y + (1-\alpha)z) + b^1) \\ &= f(\alpha y + (1-\alpha)z, \alpha(By + b^1) + (1-\alpha)(Bz + b^1)). \end{aligned}$$

We denote

$$x^1 = (y, By + b^1) \in \mathbb{R}^n, \quad x^2 = (z, Bz + b^1) \in \mathbb{R}^n.$$

Then we have

$$(\alpha y + (1-\alpha)z, \alpha(By + b^1) + (1-\alpha)(Bz + b^1)) = \alpha x^1 + (1-\alpha)x^2,$$

and consequently the convexity of the function f implies that

$$\begin{aligned} h(\alpha y + (1-\alpha)z) &= f(\alpha x^1 + (1-\alpha)x^2) \\ &\leq \alpha f(x^1) + (1-\alpha) f(x^2) \\ &= \alpha f(y, By + b^1) + (1-\alpha) f(z, Bz + b^1) \\ &= \alpha h(y) + (1-\alpha) h(z). \end{aligned}$$

The fact that the function h is proper is straightforward. □

We take any $x \in X$. It is clear that the cone of feasible directions at the point $x \in X$ can be expressed as follows:

$$K(x) = \{g \in \mathbb{R}^n : Ag = 0\}.$$

We consider the following unconstrained minimization problem:

$$\text{minimize} \quad h(y) \quad \text{subject to} \quad y \in \mathbb{R}^{n-m}. \tag{15}$$

PROPOSITION 4.2

1) Let $x^* \in X$ be a solution to problem (13)-(14). Then there exists $y^* \in \mathbb{R}^{n-m}$ such that $x^* = (y^*, By^* + b^1)$ and y^* is a solution to problem (15).
2) Let $y^* \in \mathbb{R}^{n-m}$ be a solution to problem (15). Then $x^* = (y^*, By^* + b^1)$ is a solution to problem (13)-(14).

Proof 1) Since the function f is a proper convex it is directionally differentiable. Then the function h is also directionally differentiable on $\mathbb{R}^{n-m}$ and for any $y, e \in R^{n-m}, e \neq 0$

$$\begin{aligned} h'(y,e) &= \lim_{\alpha \to +0} \frac{h(y+\alpha e) - h(y)}{\alpha} \\ &= \lim_{\alpha \to +0} \frac{f(y+\alpha e, B(y+\alpha e) + b^1) - f(y, By + b^1)}{\alpha} \\ &= \lim_{\alpha \to +0} \frac{f(y+\alpha e, By + b^1 + \alpha Be) - f(y, By + b^1)}{\alpha}. \end{aligned}$$

We denote

$$x = (y, By + b^1), \quad g = (e, Be) \in \mathbb{R}^n.$$

Then

$$\begin{aligned} h'(y,e) &= \lim_{\alpha \to +0} \frac{f(x+\alpha g) - f(x)}{\alpha} \\ &= f'(x,g). \end{aligned}$$

Let $x^* \in X$ be a solution to problem (13)-(14). Let $y^* = x^*_B \in \mathbb{R}^{n-m}$ be a vector of basic variables. It is clear that $x^* = (y^*, By^* + b^1)$. It follows from a necessary condition for a minimum that

$$f'(x^*, g) \geq 0 \quad \text{for any} \quad g \in K(x).$$

We take any direction $e \in \mathbb{R}^{n-m}$. Then $g = (e, Be) \in K(x)$ for any $x \in X$.

Indeed

$$\begin{aligned} Ag &= (A_1, A_2)(e, Be)^T \\ &= (A_1 e + A_2(Be)) \\ &= (A_1 e + A_2(-A_2^{-1} A_1 e)) \\ &= 0, \end{aligned}$$

that is $g \in K(x)$. Thus for any direction $e \in \mathbb{R}^{n-m}$ at the point y^* we have

$$h'(y^*, e) = f'(x^*, (e, Be)) \geq 0.$$

Since h is a convex function the latter means that y^* is a solution to problem (15).

2) Let y^* be a solution to problem (15). It is clear that $x^* = (y^*, By^* + b^1) \in X$. First we have to prove that for any $g \in K(x)$ there exists $e \in \mathbb{R}^{n-m}$ such that $g = (e, Be)$. Since $g \in K(x)$ it follows that $Ag = 0$. We denote by $g_B \in \mathbb{R}^{n-m}$ a vector which contains basic variables and by $g_N \in \mathbb{R}^m$ a vector which contains non-basic variables. Then we have

$$(A_1, A_2)(g_B, g_N)^T = 0,$$

and therefore

$$g_N = -A_2^{-1} A_1 g_B = B g_B.$$

Let $e = g_B$. Then

$$g = (e, Be).$$

It follows from the necessary condition for a minimum that

$$h'(y^*, e) \geq 0 \quad \text{for any} \quad e \in \mathbb{R}^{n-m}.$$

For any $g \in K(x^*)$ there exists $e \in \mathbb{R}^{n-m}$ such that

$$f'(x^*, g) = h'(y^*, e) \geq 0.$$

Since f is a convex function, x^* is a solution to problem (13)-(14).

□

Thus problem (13)-(14) can be reduced to the unconstrained minimization problem (15). This is a non-smooth convex optimization problem and we apply the discrete gradient method to solve it. A brief description of the discrete gradient method is given in the next section.

5 Discrete gradient method

5.1 *Definition of the discrete gradient*

Let φ be a locally Lipschitz continuous function defined on $\mathbb{R}^n$. Let

$$S_1 = \{g \in \mathbb{R}^n : \|g\| = 1\},\ \ G = \{e \in \mathbb{R}^n : e = (e_1, \ldots, e_n), |e_j| = 1,\ j = 1, \ldots, n\},$$

$$P = \{z : R \to \mathbb{R} :\ z(\lambda) > 0,\ \lambda > 0,\ \lambda^{-1} z(\lambda) \downarrow 0,\ \text{if}\ \ \lambda \downarrow 0\},$$

$$I(g, \alpha) = \{i \in \{1, \ldots, n\} :\ |g_i| \geq \alpha\},$$

where $\alpha \in (0, n^{-1/2}]$ is a fixed number.

Here S_1 is the unit sphere, G is the set of vertices of the unit hypercube in $\mathbb{R}^n$ and P is the set of univariate positive infinitesimal functions.

We take any $x \in \mathbb{R}^n, g \in S_1,\ i \in I(g, \alpha)$ and define a point

$$x_i^0 = x + \lambda g.$$

Let $e(\beta) = (\beta e_1, \beta^2 e_2, \ldots, \beta^n e_n)$, where $e \in G,\ \beta \in (0, 1]$. We define a sequence of n points as follows:

$$\begin{array}{lll}
x_i^1 = & x_i^0 + & (h_1, 0, \ldots, 0) \\
x_i^2 = & x_i^1 + & (0, h_2, 0, \ldots, 0) \\
x_i^3 = & x_i^2 + & (0, 0, h_3, \ldots, 0) \\
\cdot & \cdot & \cdot \\
x_i^{i-1} = & x_i^{i-2} + & (0, \ldots, 0, h_{i-1}, 0, \ldots, 0) \\
x_i^{i+1} = & x_i^{i-1} + & (0, \ldots, 0, 0, 0, h_{i+1}, 0, \ldots, 0) \\
x_i^{i+2} = & x_i^{i+1} + & (0, \ldots, 0, 0, 0, 0, h_{i+2}, 0, \ldots, 0) \\
\cdot & \cdot & \cdot \\
x_i^n = & x_i^{n-1} + & (0, \ldots, 0, h_n)
\end{array}$$

where $h_j = -z(\lambda) e_j(\beta),\ e_j(\beta) = \beta^j e_j,\ z \in P.$

Definition 5.1 (see [12, 14]) The discrete gradient of the function φ at the point $x \in \mathbb{R}^n$ is the vector $\Gamma^i(x, g, e, z, \lambda, \beta) = (\Gamma_1^i, \ldots, \Gamma_n^i) \in \mathbb{R}^n, g \in S_1, i \in I(g, \alpha)$, with the following coordinates:

$$\Gamma_j^i = [z(\lambda)e_j(\beta)]^{-1}\left[\varphi(x_i^{j-1}) - \varphi(x_i^j)\right], \quad j = 1, \ldots, n, j \neq i,$$

$$\Gamma_i^i = (\lambda g_i)^{-1}\left[\varphi(x_i^n) - \varphi(x) - \sum_{j=1, j\neq i}^{n} \Gamma_j^i(\lambda g_j - z(\lambda)e_j(\beta))\right].$$

A more detailed description of the discrete gradient and examples can be found in [12, 13].

Remark 1 It follows from Definition 5.1 that for the calculation of the discrete gradient $\Gamma^i(x, g, e, z, \lambda, \beta), i \in I(g, \alpha)$ we define a sequence of points

$$x_i^0, \ldots, x_i^{i-1}, x_i^{i+1}, \ldots, x_i^n.$$

The calculation of the discrete gradient involves evaluating the objective function φ at each point of this sequence.

Remark 2 The discrete gradient is defined with respect to a given direction $g \in S_1$. We can see that to calculate one discrete gradient we need to calculate $(n + 1)$ values of the function φ: at the point x and at the points $x_i^0, \ldots, x_i^{i-1}, x_i^{i+1}, \ldots, x_i^n$, $i \in I(g, \alpha)$. To calculate another discrete gradient at the same point x with respect to another direction $g^1 \in S_1$ we have to calculate this function n times, because we have already calculated φ at the point x.

5.2 *Calculation of the descent direction*

We consider the following unconstrained minimization problem:

$$\text{minimize} \;\; \varphi(x) \;\; \text{subject to} \;\; x \in \mathbb{R}^n \tag{16}$$

where the function φ is assumed to be Lipschitz continuous.

In the core of the discrete gradient method lies calculation of a direction of descent of the objective function φ. Therefore, we first describe an algorithm for the calculation of this direction.

Let $z \in P, \lambda > 0, \beta \in (0,1]$, the number $c \in (0,1)$ and a tolerance $\delta > 0$ be given.

Algorithm 1 Calculation of a direction of descent.

Input: Point $x \in \mathbb{R}^n$, the objective function φ.

Step 1. Choose any $g^1 \in S_1, e \in G, i \in I(g^1, \alpha)$ and compute a discrete gradient $v^1 = \Gamma^i(x, g^1, e, z, \lambda, \beta)$. Set $\overline{D}_1(x) = \{v^1\}$ and $k = 1$.

Step 2. Calculate the vector $\|w^k\|^2 = \min\{\|w\|^2 : w \in \overline{D}_k(x)\}$. If

$$\|w^k\| \leq \delta, \tag{17}$$

then stop. Otherwise go to Step 3.

Step 3. Calculate the search direction by $g^{k+1} = -\|w^k\|^{-1} w^k$.
Step 4. If

$$\varphi(x + \lambda g^{k+1}) - \varphi(x) \leq -c\lambda\|w^k\|, \tag{18}$$

then stop. Otherwise go to Step 5.

Step 5. Calculate the discrete gradient

$$v^{k+1} = \Gamma^i(x, g^{k+1}, e, z, \lambda, \beta), \ \ i \in I(g^{k+1}, \alpha),$$

construct the set $\overline{D}_{k+1}(x) = \mathrm{co}\,\{\overline{D}_k(x) \bigcup \{v^{k+1}\}\}$, set $k = k+1$ and go to Step 2.

We give some explanations to Algorithm *1*. In Step 1 we calculate the first discrete gradient with respect to an initial direction $g^1 \in \mathbb{R}^n$. The distance between the convex hull $\overline{D}_k$ of all calculated discrete gradients and the origin is calculated in Step 2. This problem can be solved using Wolfe's algorithm [15], which is very efficient numerically. If this distance is less than the tolerance $\delta > 0$ then we accept the point x as an approximate stationary point (Step 2), otherwise we calculate another search direction in Step 3. In Step 4 we check whether this direction is a direction of descent. If it is then we stop, otherwise we calculate another discrete gradient with respect to this direction in Step 5 and update the set $\overline{D}_k$. At each iteration k we improve the approximation $\overline{D}_k$ of the subdifferential of the function φ.

It is proved that Algorithm *1* is terminating (see [13]).

5.3 *Minimization algorithm*

Now we can describe the discrete gradient algorithm. Let sequences $\delta_k > 0,\ z_k \in P,\ \lambda_k > 0,\ \beta_k \in (0,1],\ \delta_k \to +0,\ z_k \to +0,\ \lambda_k \to +0,\ \beta_k \to +0,\ k \to +\infty$ and numbers $c_1 \in (0,1), c_2 \in (0,c_1]$ be given.

Algorithm 2 Discrete gradient method.

Step 1. Choose any starting point $x^0 \in \mathbb{R}^n$ and set $k = 0$.

Step 2. Set $s = 0$ and $x_s^k = x^k$.

Step 3. Apply Algorithm *1* to calculate the descent direction at $x = x_s^k, \delta = \delta_k, z = z_k, \lambda = \lambda_k, \beta = \beta_k, c = c_1$. This algorithm terminates after a finite number of iterations $l > 0$. As a result we get the set $\overline{D}_l(x_s^k)$ and an element v_s^k such that

$$\|v_s^k\|^2 = \min\{\|v\|^2 : v \in \overline{D}_l(x_s^k)\}.$$

Furthermore either $\|v_s^k\| \le \delta_k$ or for the search direction $g_s^k = -\|v_s^k\|^{-1} v_s^k$

$$\varphi(x_s^k + \lambda_k g_s^k) - \varphi(x_s^k) \le -c_1 \lambda_k \|v_s^k\|. \tag{19}$$

Step 4. If

$$\|v_s^k\| \le \delta_k \tag{20}$$

then set $x^{k+1} = x_s^k, k = k+1$ and go to Step 2. Otherwise go to Step 5.

Step 5. Construct the following iteration $x_{s+1}^k = x_s^k + \sigma_s g_s^k$, where σ_s is defined as follows

$$\sigma_s = \arg\max\left\{\sigma \ge 0 :\ \varphi(x_s^k + \sigma g_s^k) - \varphi(x_s^k) \le -c_2 \sigma \|v_s^k\|\right\}.$$

Step 6. Set $s = s+1$ and go to Step 3.

For the point $x^0 \in \mathbb{R}^n$ we consider the set $M(x^0) = \{x \in \mathbb{R}^n : \varphi(x) \le \varphi(x^0)\}$. Assume that the function φ is convex. Denote by $\partial\varphi(x)$ Clarke's subdifferential.

THEOREM 5.2 *[13] Assume that the set $M(x^0)$ is bounded for starting points $x^0 \in \mathbb{R}^n$. Then every accumulation point of $\{x^k\}$ belongs to the set $X^0 = \{x \in \mathbb{R}^n : 0 \in \partial\varphi(x)\}$.*

Table 1. The mean monthly returns of a test portfolio.

Instrument	S&P	Gov. bond	Small cap
Mean return	0.0101110	0.0043532	0.0137058

Table 2. Test portfolio covariance matrix.

	S&P	Gov. bond	Small cap
S&P	0.00324625	0.00022983	0.00420395
Gov. bond	0.00022983	0.00049937	0.00019247
Small cap	0.00420395	0.00019247	0.00764097

6 Performance of the algorithms

The main goal of our study is to establish the most computationally efficient way to solve problems A and B (Eqs.(9) and (10)). We considered the following alternatives:

(I) Convert Problem A (B) to a linear programming problem A' (B') and solve it using the simplex method;
(II) Solve problem A' (B') using the interior point method;
(III) Solve the dual to problem A' (B') using the simplex method;
(IV) Solve the dual to problem A' (B') using the interior point method;
(V) Solve problem A (B) using the discrete gradient (DG) method from [6], using a penalty function to handle the constraints;
(VI) Solve problem A (B) using the discrete gradient method, and handle the equality constraints as described in section 4.

The difference between options (V) and (VI) lies in how the linear equality constraints are handled by the algorithm: by using penalty functions in (V) and by using the approach in section 4 in (VI). The linear inequality and box constraints are dealt with using exact penalty functions in both cases.

We used several test problems to compare the above mentioned alternative algorithms. In all cases we used linear portfolios, as this is a basic requirement for options (I)-(IV). The methods (V), (VI) do not require the portfolio to be linear.

The first problem is taken from [4]. It involves a small portfolio with three positions, S&P, Government bonds and small capital stocks. It is assumed that the variables x are normally distributed, with the vector of means and covariance matrix given in Tables 1, 2. For problem A (A') the weights were fixed at $w = (0.452013, 0.115573, 0.432414)$. For problems B (B') the weights were variables, and the feasible set D was defined by the restriction $w^t E(x) \geq 0.011$.

Table 3. VaR and CVaR obtained with the minimum variance approach.

	$\beta = 0.90$	$\beta = 0.95$	$\beta = 0.99$
VaR	0.067847	0.090200	0.132128
CVaR	0.096975	0.115908	0.152977

The specific reason for choosing this test problem is that its exact solution is known. It can be obtained as the solution to the minimal variance problem [16]

$$\min_{x \in S \cap D} \sigma^2(w). \tag{21}$$

Rockafellar and Uryasev [4] prove that if the components of x are normally distributed, $\beta \geq 0.5$, and the constraint $w^t E(x) \geq R$ is active at solutions of problem B and that of (21), the solutions of the two problems coincide. Thus we have a benchmark problem to test the correctness of our algorithms. By solving (21) we obtain the optimal weights $w^* = (0.452013, 0.115573, 0.432414)$ and the values of VaR and CVaR, given in Table 3.

The second series of test problems we considered was aimed at establishing the numerical performance and scalability of the algorithms. We took linear portfolios of size 5, 10, 20, and 40, with normally distributed losses, with means randomly chosen in $[0, 0.05]$, and randomly chosen covariance matrices. We included suitable inequality constraints into the problem, notably the constraint on the smallest mean return R. We repeated the experiments 10 times and report the mean running time for each algorithm.

In the third series of test problems, we again took linear portfolios of varying size, but did not use normally distributed losses. Instead we used the generalized hyperbolic distribution [17], which is gaining popularity in financial modeling [18, 19]. The density of this distribution is given as

$$\rho_{GH}(x; \lambda, \alpha, \beta, \delta, \mu) = \frac{(a/\delta)^\lambda}{\sqrt{2\pi} K_\lambda(a\delta)} \frac{K_{\lambda - 1/2}(\alpha\sqrt{\delta^2 + (x-\mu)^2})}{(\sqrt{\delta^2 + (x-\mu)^2}/\alpha)^{1/2-\lambda}} \cdot e^{\beta(x-\mu)},$$

where $a^2 = \alpha^2 - \beta^2$ and K_λ is the modified Bessel function of the third kind with index λ. The parameters μ and δ control density's location and scale, whereas α and β play the role of the kurtosis and skewness of the distribution. This family of distributions has semi-heavy tails and is rich enough to model financial time series [18].

In our study we used parameters randomly chosen in the intervals $\lambda \in [-2, 2]$, $\alpha \in [10, 200]$,$\beta \in [-5, 5]$,$\delta \in [0.001, 0.05]$,$\mu \in [-0.001, 0.001]$, as it appears from the literature that estimation of unknown parameters from empirical data using maximum likelihood approach lead to parameters in the above ranges [20].

We used the following software packages for solving linear programming problems A' and B': `lpsolve` [21], and `glpk` [22], both implemented in C language. We used our own implementation of the discrete gradient methods in C++ language. Since our main goal was limited to establishing feasibility of non-smooth optimization techniques for portfolio optimization, rather than comparing these techniques among themselves, we did not use alternative methods (see, e.g. [23]). The computations were performed on a cluster of linux workstations, each with Xeon 2.8 GHz processor and 1 GB of RAM.

We summarize the computational results in Tables 4-10. From the Tables 4,5 it is clear that converting problems A' and B' to their duals brings a significant computational advantage. It is revealed that the interior point method is either significantly slower, or fails for this type of problems (either insufficient RAM or no convergence). There were no substantial differences between the implementations of the revised simplex method in `lpsolve` and `glpk` packages.

For the second and third series of test problems, solving the primal problems A' and B' proved to be extremely inefficient, and so was the interior point algorithm. Therefore we only report on the most efficient technique for solving the duals to A' and B', which was the revised simplex method.

Tables 6,7, reveal that when the number of scenarios q increases, the computing time for the simplex method grows at a superlinear rate. This is of course not unexpected, as simplex is an algorithm with exponential complexity, even though on average its computing time increases polynomially.

In contrast, the discrete gradient method exhibits a linear growth in complexity. The dimension of the problems A and B is relatively low, and even though the objective function becomes more complicated with the increasing q, it appears that DG does not require a significantly increased number of iterations. We attribute the increased computing time to the number of terms in the objective function. There were no substantial differences in computing time between the test problems with normally and hyperbolically distributed losses (Tables 7,8).

Our series of numerical experiments confirms the superior efficiency of the discrete gradient method as a method of calculation of VaR and CVaR, and portfolio optimization for moderate size portfolios and large number of scenarios. Despite that the linear programming formulation seems to be more attractive, it does not scale well. On the other hand, linear programming formulation will bring advantages for very large portfolios (up to 10^6 instruments), but with fewer scenarios.

Note that for an accurate approximation of the integral in (8), one needs to take a very large number of scenarios. Currently up to $q = 10^5$ scenarios are used in practical computations, but we think this number is insufficient for quality approximation. This number should be even higher for larger portfolios. Non-smooth optimization methods scale very well with the number of scenar-

Table 4. Computing time (s) for methods (I)-(VI) for Test portfolio 1 with fixed weights, as a function of the number of scenarios.

β	q	(I)	(II)	(III)	(IV)	(V)	(VI)
0.90	1000	0.01	33	0.01	0.01	0.01	0.01
0.90	20000	11	fail	2	182	0.01	0.01
0.90	40000	46	fail	8	844	1	1
0.90	100000	288	fail	14	5561	1	3
0.90	500000	7238	fail	1097	$>$ 14h	9	20
0.90	1000000	28516	fail	4392	$>$ 14h	15	45
0.95	1000	0.01	39	0.01	0.01	0.01	0.01
0.95	20000	5	fail	1	216	0.1	0.01
0.95	40000	22	fail	4	1070	0.1	1
0.95	100000	147	fail	23	6603	2	2
0.95	500000	3716	fail	555	$>$ 14h	9	15
0.95	1000000	$>$ 14h	fail	2215	$>$ 14h	19	38
0.99	1000	0.01	44	0.01	0.01	0.01	0.01
0.99	20000	2	fail	0.1	380	0.01	0.01
0.99	40000	5	fail	1	1727	1	1
0.99	100000	28	fail	5	11310	1	2
0.99	500000	674	fail	113	$>$ 14h	8	12
0.99	1000000	$>$ 14h	fail	448	$>$ 14h	18	25

ios, as this number has no effect on the number of variables. Thus non-smooth optimization offers an advantageous computational alternative for modest size portfolios.

On the other hand, problems A and B are more general than A' and B', as they do not assume linear portfolio. DG will work equally well with convex portfolios. We remind that in this case $\hat{F}_\beta$ is also a convex function of w and α.

If $L(w, x)$ is not convex but concave (as is frequently the case in practice), $\hat{F}_\beta$ is not convex, and it may possess many local minima. In this setting DG method will bring another advantage – its ability to skip shallow local minima and converge to a sufficiently deep local, if not the global minimum. We have confirmed this feature experimentally in [24], where we used DG for optimization of a very complicated objective function with myriads of local minima (the Lennard-Jones cluster problem). We demonstrated that DG systematically converges to a much deeper local minimum than an alternative local descent method.

Table 5. Computing time (s) for methods (I)-(VI) for Test portfolio 1 with variable weights, as a function of the number of scenarios.

β	q	(I)	(II)	(III)	(IV)	(V)	(VI)
0.90	1000	0.01	46	0.01	1	0.01	1
0.90	20000	40	fail	26	fail	2	1
0.90	40000	250	fail	130	fail	7	2
0.90	100000	4687	fail	914	fail	13	7
0.90	500000	43900	fail	24622	fail	60	44
0.90	1000000	$>$ 14h	fail	$>$ 14h	fail	137	68
0.95	1000	0.01	50	0.01	1	0.01	0.01
0.95	20000	21	fail	11	fail	7	1
0.95	40000	129	fail	62	fail	11	2
0.95	100000	2223	fail	467	fail	64	4
0.95	500000	$>$ 14h	fail	10014	fail	163	21
0.95	1000000	$>$ 14h	fail	$>$ 14h	fail	322	49
0.99	1000	0.01	49	0.01	1	0.01	0.01
0.99	20000	5	fail	3	fail	2	1
0.99	40000	25	fail	12	fail	14	1
0.99	100000	452	fail	72	fail	24	6
0.99	500000	$>$ 14h	fail	1651	fail	119	16
0.99	1000000	$>$ 14h	fail	$>$ 14h	fail	246	27

Table 6. Computing time (s) for methods (III),(V) and (VI) for a series of test portfolio 2 with variable weights, as a function of the number of scenarios q and the number of securities n. $\beta = 0.95$.

n	q	(III)	(V)	(VI)
3	10^3	0.028	0.01	1
3	10^4	3.05	3	2
3	10^5	290	64	4
3	10^6	fail	322	49
5	10^3	0.034	0.59	0.5
5	10^4	3.35	4	5
5	10^5	282	41	65
7	10^3	0.033	1.07	0.8
7	10^4	2.56	5.8	7
7	10^5	207	100	45
9	10^3	0.039	2.25	1.8
9	10^4	2.48	75	12
9	10^5	195	160	50
11	10^3	0.038	2.8	2.1
11	10^4	2.11	85	16
11	10^5	149	210	68

7 Parallelization

The use of computer clusters as an alternative to supercomputers is now widespread. It involves using a number of low-cost workstations interconnected by a fast network, and dividing the computational effort among the workstations. Parallelization of the computational algorithm is an important task when porting the software to computer clusters. Not all algorithms can be parallelized, and when they can, often this task involves a significant development

Table 7. Computing time (s) for methods of non-smooth optimization (V) and (VI) for a series of test portfolio 2 with variable weights, as a function of the number of scenarios q and the number of securities n. $\beta = 0.95$.

n	q	(V)	(VI)
5	10^4	4	5
5	10^5	41	65
5	10^6	558	197
5	10^7	5552	1497
10	10^4	82	12
10	10^5	166	35
10	10^6	1760	702
10	10^7	16100	5420
20	10^4	291	64
20	10^5	4555	700
20	10^6	31260	3920
20	10^7	fail	24000
40	10^4	4600	140
40	10^5	14504	1620
40	10^6	135385	16530

Table 8. Computing time (s) for methods of non-smooth optimization (V) and (VI) for a series of test portfolio 3 (generalized hyperbolic distribution) with variable weights, as a function of the number of scenarios q and the number of securities n. $\beta = 0.95$.

n	q	(V)	(VI)
5	10^4	20	2
5	10^5	159	17
5	10^6	8349	178
10	10^4	72	6
10	10^5	4668	48
10	10^6	29930	299
20	10^4	457	27
20	10^5	8943	280
20	10^6	36601	1040
40	10^4	4600	130
40	10^5	24504	1200
40	10^6	385000	11800

effort. For instance, while the original simplex method is easily parallelized, parallelization of the revised simplex method is a much harder task [25, 26].

In this section we report on parallelization of our approach to solving problems A and B using the discrete gradient method. We reported earlier on the parallelization of the discrete gradient algorithm in [24]. In this study we took a different approach of parallelizing the computation of the objective function, rather than the optimization algorithm.

The rationale for the current approach is the specifics of the problem: the

most computational effort is spent on the evaluation of the integral in (8). Therefore it makes sense to use a simpler serial version of the DG algorithm, and distribute the computation of the sum in $\hat{F}_\beta$ over different processors.

In addition, when using a large number of scenarios q, one has to store the generated values of the simulated x_i. For example for a portfolio with 100 positions and 10^7 scenarios, it requires 8×10^9 bytes ≈ 8 Gb of random access memory (RAM), which is beyond capabilities of standard workstations. When distributing computations over the nodes of a cluster, one can also use combined RAM of the nodes, by storing the subsets of the scenarios on different nodes. There is no need for much interprocessor communication, as computation of the partial sums in (8) does not require access to the values of x_i for all scenarios.

Below we outline the parallel algorithm. We use the master-slave model, in which one processor (master) controls the (serial) optimization algorithm. The optimization algorithm periodically requires computation of the values of $\hat{F}_\beta(w, \alpha)$ with given values of the arguments w, α. At this point the master processor broadcasts the vector of the arguments, and the slave processors calculate the partial sums $\sum_{i=i_k+1}^{i_{k+1}} (L(w, x_i) - \alpha)_+$, where k is the rank of the slave processor, $k = 1, \ldots, K$, and $i_1 = 0$, $i_{K+1} = q$. (The master processor $k = 1$ also performs the role of a slave processor, and calculates its share of the sum). The slave processors send their partial sums to the master, which computes the total, and subsequently the value $\hat{F}_\beta(w, \alpha)$. The values of the random variables x_i, sampled from the density $p(x)$, are computed before the start of the optimization algorithm by each processor, also in parallel, as only the subset of values $\{x_i : i = i_k + 1 \ldots, i_{k+1}\}$ is required on the processor k. This process is illustrated on Fig.1. The implementation of the parallel algorithm is straightforward.

We performed a number of numerical experiments to establish the speed-up factors when using computer clusters of various sizes. The issue here is that we still use a serial optimization algorithm, which may become a bottleneck in the computations. The efficiency of our parallelization procedure depends on the ratio between the complexity of the optimization algorithm itself and that of the computation of the objective function [27]. For a small number of variables, computation of the objective function will dominate the time spent by the optimization algorithm, and we expect a significant speed-up. For larger portfolios this may not be the case.

Tables 9,10 summarize the results of our testing. We used portfolios of different size, varied the number of scenarios q, and ran our algorithm on the clusters with the indicated number of nodes. We obtained almost a perfect linear speed-up for small and large portfolios. We measured the time spent on computation of the values of the objective function (as a proportion of the total running time), and obtained figures in $95\% - 100\%$ range. It con-

Table 9. Parallel execution of the discrete gradient method (V) for a series of test portfolios 2, $\beta = 0.95$. Computing time (s).

n	q	1 node	2 nodes	4 nodes	8 nodes	16 nodes
5	10^4	4	2	2	1	1
5	10^5	41	27	15	8	4
5	10^6	558	332	143	73	40
5	10^7	5552	2825	1260	593	466
10	10^4	40	16	12	6	4
10	10^5	166	85	60	25	13
10	10^6	1760	797	433	209	122
10	10^7	16100	7800	3912	2053	1109
20	10^4	291	99	40	21	17
20	10^5	4555	1048	285	196	96
20	10^6	31260	6958	2332	1553	826
20	10^7	fail	103061	36176	12660	6848
40	10^4	1752	2354	2733	551	410
40	10^5	14504	10697	3777	2259	1097
40	10^6	135385	58685	29187	23492	98312

Table 10. Parallel execution of the discrete gradient method (VI) for a series of test portfolios 2, $\beta = 0.95$. Computing time (s).

n	q	1 node	2 nodes	4 nodes	8 nodes	16 nodes
5	10^4	5	2	2	1	< 1
5	10^5	65	33	4	4	2
5	10^6	197	98	50	26	13
5	10^7	1497	753	390	194	97
10	10^4	12	6	6	2	1
10	10^5	35	19	9	5	3
10	10^6	702	351	183	89	45
10	10^7	5420	2780	1421	711	354
20	10^4	64	33	16	5	4
20	10^5	700	370	175	89	47
20	10^6	3920	2010	993	492	242
20	10^7	24000	11900	5867	2902	1537
40	10^4	140	72	35	19	12
40	10^5	1620	810	397	207	103
40	10^6	16530	8900	4132	2108	1044
40	10^7	fail	fail	> 14h	25964	12794

firms our initial assumption that the computational cost will be dominated by evaluation of (8), and justifies our parallelization approach.

We also note that the computational cost grows linearly with q, but it grows at a much higher rate with the number of variables n. This is not unexpected, as problems of larger size require more function evaluations to compute the discrete gradients. The cost becomes prohibitive for portfolios with 40 securities or more (on a single processor), but when executed in parallel, optimal portfolio can be found in a reasonable time on a small computer cluster. In contrast, portfolio optimization by using linear programming (problem B') is computationally infeasible.

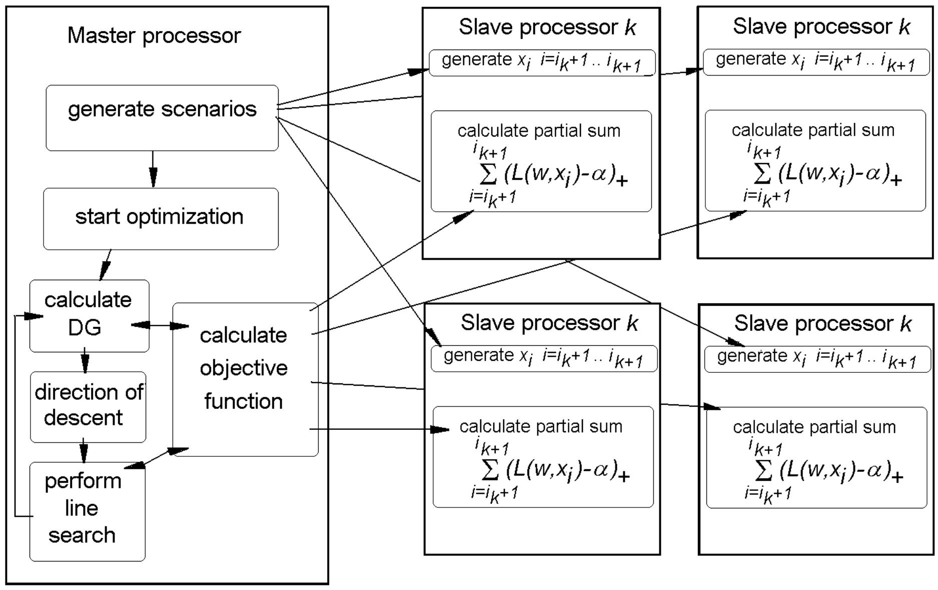

Figure 1. Parallelization of the computation of the objective function.

8 Conclusion

We have compared various techniques for computation of the Conditional Value-at-Risk and portfolio optimization with respect to this risk measure. We conclude that the use of modern non-smooth optimization methods offers significant advantages compared to linear programming approach in the cases of modest size portfolios and a large number of scenarios. These advantages include a much shorter computing time (by several orders of magnitude), lower memory requirements, greater accuracy (by using a much larger number of scenarios), straightforward parallelization of the algorithms, and an option to deal with non-linear portfolios.

We described the discrete gradient method of non-smooth optimization, and its adaptation for problems with linear constraints. Such constraints arise in portfolio optimization as balance and non-negativity constraints, and also from user requirements, such as obtaining mean return at least R. Our study shows that this method is suitable for optimization of small to medium size portfolios with respect to CVaR. We also studied its parallel execution on computer clusters, and established the effectiveness of this approach.

Acknowledgements

We wish to express our gratitude to Mr. J.E. Monsalve Tobon for his help in performing numerical experiments. We also wish to acknowledge partial support by the Victorian Partnership for Advanced Computing (VPAC), under e-Research scheme. The research by the second author was supported by the Australian Research Council. We would like to thank two anonymous referees whose valuable comments helped us improve this article.

References

[1] P. Artzner, F. Delbaen, J.-M. Eber, and D. Heath. Coherent measures of risk. *Mathematical Finance*, 9:203 –228, 1999.
[2] P. Artzner, F. Delbaen, J.-M. Eber, and D. Heath. Thinking coherently. *Risk*, 10:68–71, 1997.
[3] C. Acerbi and D. Tasche. Expected shortfall: a natural coherent alternative to value at risk. *Economic Notes*, 31:379–388, 2002.
[4] R.T. Rockafellar and S. Uryasev. Optimization of conditional value-at-risk. *Journal of Risk*, 2:21–41, 2000.
[5] R.T. Rockafellar and S. Uryasev. Conditional value-at-risk for general loss distributions. *Journal of Banking and Finance*, 26:1443–1471, 2002.
[6] A. Bagirov. A method for minimization of quasidifferentiable functions. *Optimization Methods and Software*, 17:31–60, 2002.
[7] H.-J. Lüthi and J. Doege. Convex risk measures for portfolio optimization and concepts of flexibility. *Mathematical Programming, Series B*, to appear, 2005.
[8] Y. Yamai and T. Yoshiba. On the validity of value-at-risk: Comparative analyses with expected shortfall. *Monetary and Economic Studies*, 20:57–85, 2002.
[9] H. Mausser and D. Rosen. Efficient risk/return frontiers for credit risk. *Algo Research Quarterly*, 2:35–47, 1999.
[10] F.H. Clarke. *Optimization and Nonsmooth Analysis*. John Wiley, New York, 1983.
[11] A. Bagirov. Derivative-free methods for unconstrained nonsmooth optimization and its numerical analysis. *Journal Investigacao Operacional*, 19:75–93, 1999.
[12] A. Bagirov. Continuous subdifferential approximations and their applications. *Journal of Mathematical Sciences*, 115:2567–2609, 2003.
[13] A. Bagirov. Minimization methods for one class of nonsmooth functions and calculation of semi-equilibrium prices. In A. Eberhard and et al., editors, *Progress in Optimization: Contribution from Australasia*, pages 147–175. Kluwer, Dordrecht, 1999.
[14] A. Bagirov and A. Gasanov. A method of approximating a quasidifferential. *Russian Journal of Computational Mathematics and Mathematical Physics*, 35:403–409, 1995.
[15] P.H. Wolfe. Finding the nearest point in a polytope. *Math. Progr.*, 11:128–149, 1976.
[16] H.M. Markowitz. Portfolio selection. *Journal of Finance*, 7:77–91, 1952.
[17] O. E. Barndorff-Nielsen. Exponentially decreasing distributions for the logarithm of particle size. *Proc. R. Soc. London A*, 353:401–419, 1977.
[18] E. Eberlein and U. Keller. Hyperbolic distributions in finance. *Bernoulli*, 1:281–299, 1995.
[19] C. Bauer. Value at risk using hyperbolic distributions. *J. of Economics and Business*, 52:455–467, 2000.
[20] K. Prause. *The Generalized Hyperbolic Model: Estimation, Financial Derivatives, and Risk Measures*. Phd thesis, Albert-Ludwigs Universität, 1999.
[21] P. Notebaert, K. Eikland, and M. Berkelaar. Lp_solve, http://groups.yahoo.com/group/lp_solve/, 2004.
[22] A. Makhorin. Gnu linear programming kit, http://www.gnu.org/software/glpk/glpk.html, 2004.
[23] K.C. Kiwiel. *Methods of Descent of Nondifferentiable Optimization*, volume 1133 of *Lecture Notes in Mathematics*. Springer, Berlin, 1985.
[24] G. Beliakov, A. Bagirov, and J. E. Monsalve. Parallelization of the discrete gradient method of non-smooth optimization and its applications. In *Proceedings of the 3rd International Conference on Computational Science*, volume 3 of *Springer-Verlag Lecture Notes in Computer Science (LNCS)*, pages 592–601. Springer-Verlag, Heidelberg, 2003.

[25] J. Hall. Towards a practical parallelisation of the simplex method. *Computational Management Science*, submitted, 2005.
[26] J. Hall and K. McKinnon. Asynplex, an asynchronous parallel revised simplex method algorithm. *Annals of Operations Research*, 81:27–49, 1998.
[27] G.M. Amdahl. Validity of the single-processor approach to achieving large scale computing capabilities. In *AFIPS Conference Proceeings*, volume 30, pages 483–485. AFIPS Press, 1967.