LP Algorithms for Portfolio Optimization: The PortfolioOptim Package

Andrzej Palczewski

Abstract The paper describes two algorithms for financial portfolio optimization with the following risk measures: CVaR, MAD, LSAD and dispersion CVaR. These algorithms can be applied to discrete distributions of asset returns since then the optimization problems can be reduced to linear programs. The first algorithm solves a simple recourse problem as described by Klein Haneveld and Van der Vlerk (2006) using Benders decomposition method. The second algorithm finds an optimal portfolio with the smallest distance to a given benchmark portfolio and is an adaptation of the least norm solution (called also normal solution) of linear programs due to Zhao and Li (2002). The algorithms are implemented in R in the package **PortfolioOptim**.

Introduction

The construction of an optimal portfolio of financial instruments is one of primal goals in asset management. Recent advances in risk measurement advocate using risk measures that take into account the tail behavior of asset return distributions, such as conditional value-at-risk or lower semi-absolute deviation. For these risk measures, finding optimal portfolios in risk-return setting leads to convex programming problems. For a finite discrete distribution of returns and many practically used risk measures the optimization problem can be solved by LP methods. The portfolio optimization by LP methods leads to two problems: for distributions with a large number of values (for example, a large sample from a continuous distribution) one obtains a unique solution but the LP problem is resource demanding; for distributions with a small number of values one often obtains a non-unique solution, which is of limited use for asset management. In the latter case, a unique optimal portfolio is obtained by projecting a benchmark portfolio on the whole set of solutions. In the paper we present an algorithm that solves efficiently the problem with large distributions and an algorithm that finds an orthogonal projection of the benchmark portfolio on the space of solutions.

The R language and environment for statistical computing offer a large variety of tools for portfolio optimization. General purpose optimization tools are reviewed by Theussl and Borchers (2016) (R packages for solving optimization problems) and Koenker and Mizera (2014) (R packages for convex optimization). The book by Pfaff (2016) provides an overview of specific functions for portfolio optimization which are embedded in financial packages. Here we mention a selection of R-packages dedicated primarily to portfolio optimization. The package **fPortfolio** by Würtz et al. (2009) offers a large set of functions for financial data analysis and enables portfolio optimization in mean-variance, mean-MAD and mean-CVaR settings. For these portfolio problems the package employs existing optimization tools: LP, QP and NLP solvers. The package **PortfolioAnalytics** by Peterson and Carl (2015) uses standard linear and quadratic optimization tools (**Rglpk** and **quadprog**) and a number of new packages: stochastic optimization (**DEoptim**, **GenSA**) and particle swarm optimization (**psoptim**). The package **parma** by Ghalanos (2016) offers scenario and moment based optimization of portfolios for a large class of risk and deviation measures using **Rglpk**, **quadprog** and, for non-linear optimization **nloptr**. The above mentioned packages are very effective in solving medium-size portfolio problems, however, due to their use of standard optimization tools they cannot deal with large problems either in the number of assets or the size of the distribution. Moreover, none of these packages is able to select an optimal portfolio which is closest to a given benchmark portfolio.

The new package **PortfolioOptim** overcomes the aforementioned limitations solving portfolio problems in mean-risk setting with linear portfolio constraints and risk measures that make the problem reducible to a linear program. Attempts to apply LP solvers to more general portfolio problems (cf. Mansini et al. (2014) and references therein) are not included in the package. Our first contribution is an efficient search algorithm for optimal portfolios in stochastic programs with a very large number of scenarios. A large number of scenarios in portfolio optimization appears often when a continuous distribution of returns is approximated by a discrete one. The goal is to obtain an optimal portfolio which approximates the optimal portfolio for the continuous distribution. This can be achieved by performing optimization on a large discrete sample generated from the continuous distribution under consideration. A large sample leads to a high dimensional LP problem which can be difficult to solve by general-purpose LP solvers. This problem has been addressed by Künzi-Bay and Mayer (2006) who solved the portfolio optimization problem in mean-CVaR setting using Benders decomposition. They have computed accurately CVaR of the optimal portfolio using samples of order 10^4. Our extension of this result is twofold. First, we design an algorithm that implements Benders decomposition for a general class of simple recourse problems. Portfolio optimization problems for the risk measures mentioned above are special cases of these simple recourse problems. Second, using an internal point LP solver from GLPK library and appropriately modifying the stopping criterion we substantially increase the size of discrete samples which can be used in computations. Our algorithm can perform computations for samples of order 10^6 on a standard computer in a few seconds.

Our second contribution relates to portfolio optimization problems when only a small number of random scenarios is available. Small discrete samples in portfolio optimization appear when the distribution of returns is an empirical distribution of historical returns. Usually one takes 5 or 10 years of weekly or monthly returns giving between 120 and 500 samples. Solutions to linear programs of such dimensions are in many cases non-unique and can occupy a whole face of a multidimensional simplex. In such cases, a standard software finds only one solution, usually an vertex of the solution simplex. This is often not the optimal portfolio which asset managers consider as the most appropriate. Asset managers have usually certain beliefs about the composition of optimal portfolios and those can be expressed as similarity or vicinity of the desired optimal portfolio to some benchmark portfolio. Therefore, one wants to find an optimal portfolio which has the smallest distance to a given benchmark, or, equivalently, the orthogonal projection of the benchmark on the space of optimal portfolios.

In the LP literature a special case of the above projection problem has been discussed for a long time. This is the problem of the least norm LP solution, also called normal solution (cf. Zhao and Li (2002) and references cited therein). Unfortunately, the algorithms which find normal solutions cannot be easily adapted to our problem. The reason is that portfolio weights make up only a fraction of all coordinates of the LP solution (see the examples in Sections LP computable portfolio problems and Benders decomposition). Contrary to the normal solution which is the projection of the origin onto the simplex of optimal solutions, we are looking for a projection onto a subspace of portfolio weights. In addition, we are not looking for a solution with the least norm but for a solution with the smallest distance to an arbitrary vector (benchmark portfolio). It appears however, that the regularized central path algorithm due to Zhao and Li (2002) which solves simultaneously primal and dual LP problems finding the least norm solutions to both problems, can be modified to our purposes. Our contribution is the extension of this algorithm to the following problem. Given the set S^* of optimal solutions to an LP problem, find $x^* \in S^*$ such that $\|B(x^* - \hat{x})\| \le \|B(x - \hat{x})\|$ for all $x \in S^*$, where $\hat{x}$ is a given vector and B is the operator of projection on a subspace. This general algorithm is then adapted to obtain an optimal portfolio with the smallest distance to a benchmark (see Section Projection algorithm for portfolio optimization for details).

The rest of the paper is organized as follows. In Section LP computable portfolio problems, we describe the portfolio optimization problems that can be reduced to LP problems and are considered as working examples in the rest of the paper. These portfolio optimization problems are also implemented in the package **PortfolioOptim**. Section Benders decomposition analyzes Benders decomposition algorithm applied to simple recourse problems. As an illustration, we present applications to the portfolio optimization in mean-CVaR and mean-LSAD settings. Section Projection algorithm for portfolio optimization describes the adaptation of the path-following algorithm of Zhao and Li (2002) to the construction of an optimal portfolio with the smallest distance to a benchmark portfolio. We present also the proof of convergence for the modified algorithm. Computational examples and the analysis of performance of both algorithms are presented in Section Numerical examples. The paper ends with short Summary.

LP computable portfolio problems

We consider the portfolio optimization problem in mean-risk setting. The risk is measured by a risk (or deviation) measure $\mathcal{R}$ (for the definitions of risk and deviation measures the reader is advised to consult the paper by Rockafellar et al. (2006)). Let R be a distribution of returns of J risky assets. We denote by $\hat{R}$ centered returns, i.e. $\hat{R} = R - \mathbb{E}[R]$. Consider the problem of finding an optimal portfolio u which fulfills the conditions

$$\begin{cases} \mathcal{R}\left(u^T R\right) \to \min, \\ u^T \mathbb{E}\left[R\right] \ge r_0, \\ u \in \mathbb{X}, \end{cases} \tag{1}$$

where $\mathbb{X}$ is a polyhedral set (i.e. given by linear constraints) and r_0 is the required return of the portfolio. The set $\mathbb{X}$ is usually the whole space (when no limitations on trading positions are imposed) or the positive orthant (when short-selling of assets is prohibited).

We will assume that at optimum the constraint $u^T \mathbb{E}[R] \ge r_0$ is binding. This is the case when r_0 is not smaller than the return for the minimal risk portfolio, i.e. the portfolio which solves the problem $\min_{u \in \mathbb{X}} \mathcal{R}(u^T R)$.

It appears that for a number of risk measures the optimization problem (1) formulated for a discrete distribution R can be reduced to a linear program. Let the asset returns R be given by a discrete distribution placing weights p_n at points r_n, $n = 1, \ldots, N$. When the distribution is centered points of $\hat{R}$ will be denoted by $\hat{r}_n$.

Consider the risk measured by conditional value at risk (CVaR). For a random outcome X we define $\mathrm{VaR}_\alpha(X)$ as

$$\mathrm{VaR}_\alpha(X) = -\inf\left\{z : \mathbb{P}(X \le z) > \alpha\right\}.$$

Then the conditional value at risk is defined as

$$\mathrm{CVaR}_\alpha(X) = \frac{1}{\alpha}\int_0^\alpha \mathrm{VaR}_p(X)\,dp.$$

Rockafellar and Uryasev (2000) observed that for $X = u^T R$ the minimization of CVaR can be formulated as the following nonlinear problem

$$\min_{u,\xi} \xi + \frac{1}{1-\alpha}\mathbb{E}\left[\left(-u^T R - \xi\right)^+\right], \tag{2}$$

where the latent variable ξ corresponds to $\mathrm{VaR}_\alpha(u^T R)$. For discrete distributions and the risk measured by CVaR, the portfolio optimization problem can therefore be reformulated as the linear program

$$\begin{cases} \xi + \frac{1}{1-\alpha}\sum_{n=1}^N p_n y_n \to \min, \\ y_n \ge 0, \quad n = 1, \ldots, N, \\ y_n + \xi + r_n^T u \ge 0, \quad n = 1, \ldots, N, \\ \mu^T u \ge r_0, \\ u \in \mathbb{X}. \end{cases} \tag{3}$$

Here $\mu = \mathbb{E}[R]$, while y_n plays a role of the shortfall of return $u^T r_n$ below ξ: $y_n = (-\xi - u^T r_n)^+ = (\xi + u^T r_n)^-$. For deviation CVaR we replace r_n by $\hat{r}_n$ in (3).

Consider now mean-MAD and mean-LSAD optimization problems. MAD (mean absolute deviation) is defined as (cf. Konno and Yamazaki (1991))

$$MAD\left(u^T R\right) = \mathbb{E}\left[\left|u^T R - \mathbb{E}[u^T R]\right|\right] = \mathbb{E}\left[|u^T \hat{R}|\right]. \tag{4}$$

For discrete distributions mean-MAD optimization problem reads

$$\begin{cases} \sum_{n=1}^N p_n \left|\hat{r}_n^T u\right| \to \min, \\ \mu^T u \ge r_0, \\ u \in \mathbb{X}. \end{cases} \tag{5}$$

This problem can be formulated as the linear program

$$\begin{cases} \sum_{n=1}^N p_n y_n \to \min, \\ y_n \ge 0, \quad n = 1, \ldots, N \\ y_n + \hat{r}_n^T u \ge 0, \quad n = 1, \ldots, N, \\ y_n - \hat{r}_n^T u \ge 0, \quad n = 1, \ldots, N, \\ \mu^T u \ge r_0, \\ u \in \mathbb{X}. \end{cases} \tag{6}$$

For LSAD (lower semi absolute deviation) as defined by Konno (1990) or Konno et al. (2002)

$$LSAD\left(u^T R\right) = \mathbb{E}\left[\left|u^T R - \mathbb{E}[u^T R]\right|_-\right] = \mathbb{E}\left[|u^T \hat{R}|_-\right], \tag{7}$$

mean-LSAD optimization has the form

$$\begin{cases} \sum_{n=1}^N p_n \left|\hat{r}_n^T u\right|_- \to \min, \\ \mu^T u \ge r_0, \\ u \in \mathbb{X}. \end{cases} \tag{8}$$

This leads to the following linear program

$$\begin{cases} \sum_{n=1}^{N} p_n y_n \to \min, \\ y_n \geq 0, \quad n = 1,\ldots,N, \\ y_n + \hat{r}_n^T u \geq 0, \quad n = 1,\ldots,N, \\ \mu^T u \geq r_0, \\ u \in \mathbb{X}. \end{cases} \tag{9}$$

Since $LSAD(u^T R) = \frac{1}{2} MAD(u^T R)$, in what follows we shall consider only one of the above optimization problems.

Benders decomposition

Algorithm

In this section, we present a solution to a simple linear recourse problem with random technology matrix when the distribution of stochastic variable is represented by a large number of scenarios. The simple recourse problem is a special case of two-stage recourse problem

$$\begin{cases} c^T x + Q(x) \to \min, \\ x \in \mathbb{X}, \end{cases} \tag{10}$$

with the following form of the recourse subproblem

$$\begin{cases} Q(x) = \mathbb{E}\left[v(x)\right], \\ v(x) = \min\limits_{y} \left\{ \left(q^+\right)^T y^+ + \left(q^-\right)^T y^- : y^+ - y^- = b - Ax,\ y^+, y^- \in \mathbb{R}_+^m \right\}. \end{cases} \tag{11}$$

In this framework, q^+ and q^- are known penalty costs, matrix $A = A(\omega)$ and vector $b = b(\omega)$ are random and $\mathbb{X}$ is a bounded, convex subset in $\mathbb{R}^l$.

We see that each pair of recourse variables (y_i^+, y_i^-), $i = 1,\ldots,m$, depends only on the i-th row in the condition $b - Ax$, so that their optimal values can be determined independently. Thus, the second-stage value function $v(x)$ is separated into the sum of m functions

$$v(x) = \sum_{i=1}^{m} \min_{y_i} \left\{ q_i^+ y_i^+ + q_i^- y_i^- : y_i^+ - y_i^- = b_i - A_i x,\ y_i^+, y_i^- \geq 0 \right\},$$

where A_i is the ith row of matrix A and b_i is the ith element of vector b.

For each optimization problem

$$\min_{y_i} \left\{ q_i^+ y_i^+ + q_i^- y_i^- : y_i^+ - y_i^- = b_i - A_i x,\ y_i^+, y_i^- \geq 0 \right\}, \tag{12}$$

the dual problem has the form

$$\sup_{\lambda_i} \left\{ \lambda_i \left(b_i - A_i x\right) : \lambda_i \leq q_i^+, -\lambda_i \leq q_i^- \right\}.$$

The dual problem is feasible only if $q_i^+ + q_i^- \geq 0$. When this condition holds, the solution to the dual problem is given by

$$\lambda_i = \begin{cases} q_i^+, & \text{if } b_i - A_i x > 0, \\ -q_i^-, & \text{if } b_i - A_i x \leq 0, \end{cases}$$

and the optimal solution to problem (12) has the form

$$\begin{aligned} y_i^+ &= \max\left(0, \left(b_i - A_i x\right)\right), \\ y_i^- &= \max\left(0, -\left(b_i - A_i x\right)\right). \end{aligned}$$

Then the value function $Q(x)$ can be written as

$$\begin{aligned} Q(x) &= \sum_{i=1}^{m} \left(q_i^+ \mathbb{E}\left[\left(b_i - A_i x\right)^+\right] + q_i^- \mathbb{E}\left[\left(b_i - A_i x\right)^-\right] \right) \\ &= \sum_{i=1}^{m} \left(q_i^+ \mathbb{E}\left[\left(A_i x - b_i\right)^-\right] + q_i^- \mathbb{E}\left[\left(A_i x - b_i\right)^+\right] \right). \end{aligned} \tag{13}$$

Since $s^+ - s^- = s$, we have

$$\begin{aligned} Q(x) &= \sum_{i=1}^{m} \left(q_i^- \left(\bar{A}_i x - \bar{b}_i \right) + \left(q_i^+ + q_i^- \right) \mathbb{E} \left[\left(A_i x - b_i \right)^- \right] \right) \\ &= \left(q^- \right)^T \bar{A} x - \left(q^- \right)^T \bar{b} + \mathbb{E} \left[\left(q^+ + q^- \right)^T \left(b - Ax \right)^+ \right], \end{aligned} \tag{14}$$

where $\bar{A}_i = \mathbb{E}[A_i]$ and $\bar{b}_i = \mathbb{E}[b_i]$.

For a discrete probability space with $\mathbb{P}(\omega = \omega_n) = p_n$, $n = 1, \dots, N$, the simple recourse problem can be reformulated in the following way

$$\begin{cases} c^T x + \left(q^- \right)^T \left(\bar{A} x - \bar{b} \right) + \sum_{n=1}^{N} p_n h_n \to \min, \\ h_n \geq \left(q^+ + q^- \right)^T \left(b(\omega_n) - A(\omega_n) x \right), \\ h_n \geq 0, \\ x \in \mathbb{X}. \end{cases} \tag{15}$$

This linear program with a large size N of the probability space is difficult to solve due to the number of constraints. Klein Haneveld and Van der Vlerk (2006) solved it using a special version of the L-shaped method. Their approach was further extended by Künzi-Bay and Mayer (2006) who applied directly Benders decomposition (cf., Benders (1962)). In terms of Benders cuts (15) is equivalent to (we skip $(q^-)^T \bar{b}$ which is constant)

$$\begin{cases} \min_{x,w} c^T x + \left(q^- \right)^T \bar{A} x + w, \\ w \geq \sum_{n \in K} p_n \left(q^+ + q^- \right)^T \left(b(\omega_n) - A(\omega_n) x \right), \quad \text{for all } K \subset \mathcal{M}, \\ w \geq 0, \\ x \in \mathbb{X}, \end{cases} \tag{16}$$

where $\mathcal{M} = \{1, \dots, N\}$.

This problem is even harder than the original linear program (15) as there are 2^N constraints, but Benders (1962) showed that the above problem can be solved through a sequence of expanding relaxed problems

$$\begin{cases} \min_{x,w} c^T x + \left(q^- \right)^T \bar{A} x + w, \\ w \geq \sum_{n \in K_k} p_n \left(q^+ + q^- \right)^T \left(b(\omega_n) - A(\omega_n) x \right), \quad k = 1, \dots, \nu, \\ w \geq 0, \\ x \in \mathbb{X}, \end{cases} \tag{17}$$

ν is the number of steps, $K_k \subset \mathcal{M}$ are constraints added in step k with $K_k \neq K_l$ when $k \neq l$. Hence, each successive relaxed problem adds more constraints to those already present in the previous steps.

Let

$$\eta_k = \sum_{n \in K_k} p_n \left(q^+ + q^- \right)^T A(\omega_n), \quad \zeta_k = \sum_{n \in K_k} p_n \left(q^+ + q^- \right)^T b(\omega_n).$$

Then (17) is written as the linear program

$$\begin{cases} \min_{x,w} c^T x + \left(q^- \right)^T \bar{A} x + w, \\ w \geq \zeta_k - \eta_k x, \quad k = 1, \dots, \nu, \\ w \geq 0, \\ x \in \mathbb{X}. \end{cases} \tag{18}$$

The complete algorithm is as follows:

Step 1. Initialization: set $K_1 = \mathcal{M}$, $\eta_1 = \sum_{n=1}^{N} p_n (q^+ + q^-)^T A(\omega_n)$, $\zeta_1 = \sum_{n=1}^{N} p_n (q^+ + q^-)^T b(\omega_n)$ and $\nu = 1$. Choose computation accuracy ε.

Step 2. Solve problem (18) and denote the solution by (x^*, w^*). Put

$$K^* = \left\{ n \in \mathcal{M} : \left(q^+ + q^- \right)^T \left(b(\omega_n) - A(\omega_n) x^* \right) \geq 0 \right\},$$

$$w_+^* = \sum_{n \in K^*} p_n \left(q^+ + q^- \right)^T \left(b(\omega_n) - A(\omega_n) x^* \right), \quad \hat{w}^* = \zeta_\nu - \eta_\nu x^*.$$

Compute

$$\underline{F} = c^T x^* + \left(q^- \right)^T \bar{A} x^* + w_+^*, \quad \overline{F} = c^T x^* + \left(q^- \right)^T \bar{A} x^* + \hat{w}^*.$$

Step 3. If $(\overline{F} - \underline{F}) \leq \varepsilon$ then **stop**. x^* is an optimal solution.

Step 4. Set $\nu = \nu + 1$, $K_\nu = K^*$. Compute $\eta_\nu = \sum_{n \in K_\nu} p_n (q^+ + q^-)^T A(\omega_n)$ and $\zeta_\nu = \sum_{n \in K_\nu} p_n (q^+ + q^-)^T b(\omega_n)$. Add this new constraint to the set of constraints and go to Step 2.

The algorithm written as an R script can be seen on Fig. 1.

```
Set $\nu = 0$ and $K_1 = \mathcal{M}$.
Put $cmat = (c + (q^-)^T \bar{A}, 1)$, $Amat = NULL$, $bmat = NULL$

repeat
   Put $\nu = \nu + 1$
   Compute $\eta_\nu = \sum_{n \in K_\nu} p_n (q^+ + q^-)^T A(\omega_n)$ and $\zeta_\nu = \sum_{n \in K_\nu} p_n (q^+ + q^-)^T b(\omega_n)$
   Put $Amat = rbind(Amat, (-\eta_\nu, \zeta_\nu, -1))$, $bmat = c(bmat, 0)$
   Solve the linear problem $\min_{(x,w)} cmat^T * (x, w)$ with the constraints
   $Amat * (x, w)^T \leq bmat^T$ and denote its solution by $(x^*, w^*)$.
   Put
         $K^* = \left\{ n \in \mathcal{M} \colon \left(q^+ + q^-\right)^T \left(b\left(\omega_n\right) - A\left(\omega_n\right) x^*\right) \geq 0 \right\},$
         $w^*_+ = \sum_{n \in K^*} p_n \left(q^+ + q^-\right)^T \left(b\left(\omega_n\right) - A\left(\omega_n\right) x^*\right), \quad \hat{w}^* = \zeta_\nu - \eta_\nu x^*.$
   Set $K_{\nu+1} = K^*$
   Compute
         $\underline{F} = c^T x^* + \left(q^-\right)^T \bar{A} x^* + w^*_+, \quad \overline{F} = c^T x^* + \left(q^-\right)^T \bar{A} x^* + \hat{w}^*.$

while $(\overline{F} - \underline{F}) > \varepsilon$

Output: $x^*$
```

Figure 1: Benders decomposition algorithm for a simple recourse problem

Examples

Conditional value at risk (CVaR)

We begin with an example of mean-CVaR optimization for a discrete distribution of returns. Our goal is to find portfolio u which solves

$$\begin{cases} \mathrm{CVaR}_\alpha \left(u^T R\right) \to \min, \\ u^T \mathbb{E}\left[R\right] \geq r_0, \\ u \in \mathbb{X}. \end{cases} \tag{19}$$

As is shown in Section LP computable portfolio problems, this problem is reduced to the linear program

$$\begin{cases} \xi + \frac{1}{1-\alpha} \sum_{n=1}^N p_n y_n \to \min, \\ y_n \geq 0, \quad n = 1, \ldots, N, \\ y_n + \xi + r_n^T u \geq 0, \quad n = 1, \ldots, N, \\ u^T \mu \geq r_0, \\ u \in \mathbb{X}, \end{cases} \tag{20}$$

where $\mu = \mathbb{E}[R]$. Benders decomposition for the above problem leads to the sequence of expanding relaxed problems

$$\begin{cases} \min_{u,\xi,w} \xi + \frac{1}{1-\alpha} w, \\ \sum_{n\in K_k} p_n \left(\xi + r_n^T u\right) + w \geq 0, \quad k = 1,\ldots,\nu, \\ u^T \mu \geq r_0, \\ u \in \mathbb{X}. \end{cases} \tag{21}$$

Using notation from Section Benders decomposition we identify $x = (u,\xi)$, $q^- = 0$, $(q^+)^T A(\omega_n) = (r_n, 1)$, $b(\omega_n) = 0$ and $c = (0_J, 1)$, where 0_J is the zero vector of length J. The algorithm can be readily applied.

Mean absolute deviation(MAD) and lower semi-absolute deviation (LSAD)

Due to the relation $LSAD(u^T R) = \frac{1}{2} MAD(u^T R)$ it is sufficient to consider only the mean-LSAD portfolio optimization:

$$\begin{cases} \text{LSAD}\left(u^T R\right) \to \min, \\ u^T \mu \geq r_0, \\ u \in \mathbb{X}. \end{cases} \tag{22}$$

As before, for a discrete set of returns that problem can be formulated as a linear program (see equation (9)) and Benders decomposition can be applied leading to a sequence of expanding relaxed problems. Introducing the new variable

$$\eta_k = \sum_{n\in K_k} p_n \hat{r}_n,$$

we reduce the problem to the linear program

$$\begin{cases} \min_{u,w} w, \\ \eta_k^T u + w \geq 0, \quad k = 1,\ldots,\nu, \\ \mu^T u \geq r_0, \\ u \in \mathbb{X}. \end{cases} \tag{23}$$

Similarly as for CVaR, we recognize in that formulation the Benders problem from Section Benders decomposition.

Projection algorithm for portfolio optimization

Algorithm

As we have discussed earlier, some mean-risk optimization problems can be transformed into linear programs. Those linear programs can be written down in the standard form

$$\begin{aligned} & c^T x \to \min, \\ & Ax \geq b, \\ & x \geq 0, \\ & x \in \mathbb{R}^n. \end{aligned} \tag{24}$$

In practical computations, it appears that solutions to the above linear program are sometimes non-unique and form a (multidimensional) simplex. A typical linear solver find only one of the solutions – usually one of the vertices of the solution simplex. In this section our aim is to find a solution to the above linear program which is closest to a given vector. It often lies in the interior of a face of the solution simplex.

Let S^* denote the set of optimal solutions to (24), B be a given invertible matrix in $\mathbb{R}^n$ and $\hat{x}$ a vector from $\mathbb{R}^n$. We want to find $x^* \in S^*$ such that

$$\|B\left(x^* - \hat{x}\right)\| \leq \|B\left(x - \hat{x}\right)\| \text{ for all } x \in S^*. \tag{25}$$

The least norm solution to linear program mentioned in the Introduction is a special case of problem (24–25). It corresponds to $\hat{x} = 0$ and B being an identity matrix. Finding an optimal portfolio closest to a given benchmark in the framework of LP problems can also be reduced, after some modification, to the solution of problem (24–25). To describe this modification observe that for LP computable portfolio problems the independent variable x in (24) contains more coordinates that only portfolio weights. Let $x = (x', u)$ where u corresponds to portfolio weights and x' contains all other coordinates (for CVaR optimization (3) $x = (\xi, y, u)$, hence $x' = (\xi, y)$, for MAD optimization (6) and LSAD optimization (9) $x = (y, u)$). Given the benchmark portfolio $\hat{u}$, the objective is to find an optimal portfolio which minimizes the distance $\|u - \hat{u}\|$. To achieve that goal by solving problem (24–25) we have to extend $\hat{u}$ to a vector $\hat{x} = (\hat{x}', \hat{u})$ on the whole space $\mathbb{R}^n$ taking arbitrary $\hat{x}'$ and projecting the difference $(x - \hat{x})$ on the subspace spanned by the portfolio coordinates. Let B^* be this projection operator, then the aim is to minimize $\|B^*(x - \hat{x})\|$. Matrix B^* is diagonal with entries 1 on the diagonal positions corresponding to portfolio weights and 0 otherwise. Since B^* is not invertible the considered portfolio problem cannot be reduced to the solution of problem (24–25). Hence we introduce a matrix B replacing in B^* zero diagonal entries with some small positive number ε. Then B is invertible and an optimal portfolio close to a benchmark can be found by solving problem (24–25). One can further improve the accuracy of the computation by redoing the optimization with $\hat{x} = (x'^*, \hat{u})$, where x'^* is the solution obtained in the first optimization.

To solve problem (24–25) we follow the approach by Zhao and Li (2002) based on the path-following algorithm. We begin with the reformulation of (24) as the logarithmic barrier problem

$$\begin{aligned} & c^T x - \rho \left(\sum_{i=1}^{n} \log x_i + \sum_{i=1}^{m} \log z_i \right) \to \min, \\ & Ax - z = b, \\ & x, z > 0, \\ & x \in \mathbb{R}^n, z \in \mathbb{R}^m, \end{aligned} \tag{26}$$

where z is introduced to replace the inequality $Ax \geq b$ by equality and $\rho > 0$ is a regularizing parameter.

The Lagrangian for this problem reads

$$L(x, y, z) = c^T x + y^T (z - Ax + b) - \rho \left(\sum_{i=1}^{n} \log x_i + \sum_{i=1}^{m} \log z_i \right)$$

and the Kuhn-Tucker solvability conditions are

$$\begin{aligned} \operatorname{diag}(x)\, s &= \rho e, \\ \operatorname{diag}(z)\, y &= \rho e, \\ s + A^T y - c &= 0, \\ z - Ax + b &= 0, \end{aligned} \tag{27}$$

where diag(u) denotes the diagonal matrix with vector u on diagonal; the new variable $s = \rho \operatorname{diag}(x)^{-1} e$ is introduced to obtain the canonical representation of the central path approach; and $e = (1, \ldots, 1)$ (of an appropriate dimension).

To find the solution to (26) which fulfills the condition $\|B(x - \hat{x})\|^2 \to \min$ for a given vector $\hat{x}$ and matrix B, we modify the Lagrangian

$$L_P(x, y, z) = c^T x + y^T (z - Ax + b) - \rho \left(\sum_{i=1}^{n} \log x_i + \sum_{i=1}^{m} \log z_i \right) + \frac{1}{2} \theta \left(\|B(x - \hat{x})\|^2 - \|y\|^2 \right).$$

The stationary point of this Lagrangian gives the primal solution for which $\|B(x - \hat{x})\|^2$ achieves minimal value and the dual solution which has minimal norm. The Kuhn-Tucker conditions give

$$\begin{aligned} \operatorname{diag}(x)\, s &= \rho e, \\ \operatorname{diag}(z)\, y &= \rho e, \\ s + A^T y - c &= \theta B^T B (x - \hat{x}), \\ z - Ax + b &= \theta y. \end{aligned} \tag{28}$$

To simplify the problem we assume that $\theta = \kappa \rho^p$ for given constants $\kappa \in (0, 1]$ and $p \in (0, 1)$. We define the nonlinear mapping

$$F_\rho(x, y, s, z) = \begin{pmatrix} \operatorname{diag}(x)\, s - \rho e \\ \operatorname{diag}(z)\, y - \rho e \\ s + A^T y - c - \kappa \rho^p B^T B (x - \hat{x}) \\ z - Ax + b - \kappa \rho^p y \end{pmatrix}. \tag{29}$$

The problem is now to find $(x^*, y^*, s^*, z^*) \geq 0$ which solve the equation

$$F_0\left(x^*, y^*, s^*, z^*\right) = 0. \tag{30}$$

The solution to this equation is searched by the Newton iterative method

$$F_{\rho_k}\left(x_k, y_k, s_k, z_k\right) + \nabla F_{\rho_k}\left(x_k, y_k, s_k, z_k\right)\left(\Delta x, \Delta y, \Delta s, \Delta z\right) = 0, \tag{31}$$

where $\nabla F_\rho(x, y, s, z)$ denotes the gradient of function F_ρ given by the expression

$$\nabla F_\rho\left(x, y, s, z\right) = \begin{pmatrix} \operatorname{diag}(s) & 0 & \operatorname{diag}(x) & 0 \\ 0 & \operatorname{diag}(z) & 0 & \operatorname{diag}(y) \\ -\kappa\rho^p B^T B & A^T & \mathbf{1} & 0 \\ -A & -\kappa\rho^p \mathbf{1} & 0 & \mathbf{1} \end{pmatrix}. \tag{32}$$

These iterative solutions form the path-following algorithm in a neighborhood of the regularized central path

$$\mathcal{N}_\beta\left(\rho\right) = \left\{(x, y, s, z) : \left\|F_\rho\left(x, y, s, z\right)\right\|_\infty \leq \beta\rho\right\},$$

with $\beta \in (0, 1)$.

The complete algorithm is as follows:

Central path projection algorithm

Step 1. Initialization. Set $\beta \in (0, 1)$ and assign scalars b_1, b_2 and σ in $(0, 1)$. Select $(x_0, y_0, s_0, z_0) > 0$, $\kappa \in (0, 1)$, $p \in (0, 1)$ and $\rho_0 \in (1, \infty)$ such that $(x_0, y_0, s_0, z_0) \in \mathcal{N}_\beta(\rho_0)$.

Step 2. Newton's iterates. If $F_{\rho_k}(x_k, y_k, s_k, z_k) = 0$ put

$$(x_{k+1}, y_{k+1}, s_{k+1}, z_{k+1}) = (x_k, y_k, s_k, z_k)$$

and go to Step 3.
Otherwise, find $(\Delta x, \Delta y)$ which solve the following linear system

$$\begin{aligned} &\begin{pmatrix} \operatorname{diag}(s_k) + \kappa\left(\rho_k\right)^p B^T B \operatorname{diag}(x_k) & -\operatorname{diag}(x_k) A^T \\ \operatorname{diag}(y_k) A & \operatorname{diag}(z_k) + \kappa\left(\rho_k\right)^p \operatorname{diag}(y_k) \end{pmatrix} \begin{pmatrix} \Delta x \\ \Delta y \end{pmatrix} \\ &= \begin{pmatrix} \rho_k e - \operatorname{diag}(x_k) s_k \\ \rho_k e - \operatorname{diag}(z_k) y_k \end{pmatrix} - \begin{pmatrix} c \operatorname{diag}(x_k)\left(\kappa\left(\rho_k\right)^p B^T B\left(x_k - \hat{x}\right) - A^T y_k - s_k + c\right) \\ \operatorname{diag}(y_k)\left(A x_k + \kappa\left(\rho_k\right)^p y_k - z_k - b\right) \end{pmatrix}. \end{aligned} \tag{33}$$

Then set

$$\begin{aligned} \begin{pmatrix} \Delta s \\ \Delta z \end{pmatrix} &= \begin{pmatrix} \kappa\left(\rho_k\right)^p B^T B & -A^T \\ A & \kappa\left(\rho_k\right)^p \mathbf{1} \end{pmatrix} \begin{pmatrix} \Delta x \\ \Delta y \end{pmatrix} \\ &+ \begin{pmatrix} \kappa\left(\rho_k\right)^p B^T B\left(x_k - \hat{x}\right) - A^T y_k - s_k + c \\ A x_k + \kappa\left(\rho_k\right)^p y_k - z_k - b \end{pmatrix}. \end{aligned} \tag{34}$$

Find α such that $(x_k + \lambda\Delta x, y_k + \lambda\Delta y, s_k + \lambda\Delta s, z_k + \lambda\Delta z) > 0$ for all $\lambda \in (0, \alpha)$. Then find the maximal improvement step for λ by the Armijo rule: find the smallest j such that $\lambda_j = \alpha b_1^j$ and

$$\begin{aligned} &\left\|F_{\rho_k}\left(x_k + \lambda_j\Delta x, y_k + \lambda_j\Delta y, s_k + \lambda_j\Delta s, z_k + \lambda_j\Delta z\right)\right\|_\infty \\ &\leq \left(1 - \sigma\lambda_j\right)\left\|F_{\rho_k}\left(x_k, y_k, s_k, z_k\right)\right\|_\infty \end{aligned} \tag{35}$$

Set

$$(x_{k+1}, y_{k+1}, s_{k+1}, z_{k+1}) = (x_k, y_k, s_k, z_k) + \lambda_j\left(\Delta x, \Delta y, \Delta s, \Delta z\right)$$

and go to Step 3.

Step 3. Reduction of ρ. Find the maximal improvement step for ρ_k by the Armijo rule: find the smallest j such that $\gamma_j = b_2^j$ and

$$\left\|F_{(1-\gamma_j)\rho_k}\left(x_{k+1}, y_{k+1}, s_{k+1}, z_{k+1}\right)\right\|_\infty \leq \beta\left(1 - \gamma_j\right)\rho_k.$$

Set $\rho_{k+1} = (1 - \gamma_j)\rho_k$ and go to Step 2.

We prove now that starting from the initial iterate defined in Step 1, we obtain a convergent sequence of iterates. The estimate of the norm F creates a difficulty because of the term ρ_k^p which for

$\rho_k < 1$ cannot be estimated by the first power of ρ_k. That requires such a choice of the starting point which cancels the terms with ρ^p. This goal is achieved by a proper choice of constant κ. The following lemma is an adaptation of Lemma 4.1 from Zhao and Li (2002).

Lemma 2.4.1. *The central path projection algorithm is well-defined. The sequence ρ_k is monotonically decreasing and $(x_k, y_k, s_k, z_k) \in \mathcal{N}_\beta(\rho_k)$ for all $k \geq 0$.*

Proof. The proof goes along the lines of the original proof of Zhao and Li (2002). We have only to remark that the nonsingularity of matrix ∇F_ρ follows from the fact that the matrix

$$\begin{pmatrix} \kappa\rho^p B^T B & -A^T \\ A & \kappa\rho^p \mathbf{1} \end{pmatrix}$$

is positive semidefinite for $\rho > 0$.

For the correctness of Step 3 of the algorithm, we have to show that point $(x_{k+1}, y_{k+1}, s_{k+1}, z_{k+1})$ belongs to $\mathcal{N}_\beta(\rho_{k+1})$. To this end, we have to prove the estimate

$$\left\|F_{\rho_k}(x,y,s,z) - F_{\rho_l}(x,y,s,z)\right\|_\infty \leq (\rho_l - \rho_k) + \left((\rho_l)^p - (\rho_k)^p\right) \|(x - \hat{x}, y)\|_\infty ,$$

for $\rho_l \geq \rho_k$ and $(x,y,s,z) > 0$. Taking into account that $\kappa \leq 1$ that estimate follows from the definition of F

$$\begin{aligned} &\left\|F_{\rho_k}(x,y,s,z) - F_{\rho_l}(x,y,s,z)\right\|_\infty \\ &\quad\leq (\rho_l - \rho_k) + \kappa\left((\rho_l)^p - (\rho_k)^p\right)\|y\|_\infty + \kappa\left((\rho_l)^p - (\rho_k)^p\right)\left\|B^T B(x - \hat{x})\right\|_\infty \\ &\quad\leq (\rho_l - \rho_k) + \left((\rho_l)^p - (\rho_k)^p\right)\|(x - \hat{x}, y)\|_\infty . \end{aligned}$$

The rest of the proof is similar as in Zhao and Li (2002). The boundedness of (x_k, y_k) follows from Lemma 2.4.2 below. □

The proof of the convergence of the iterative sequence requires some modification of the proof by Zhao and Li (2002). The modification is formulated in the following lemma.

Lemma 2.4.2. *When the solution set to equation (30) is nonempty, then the sequence (x_k, y_k, s_k, z_k) obtained by the central path projection algorithm is bounded.*

Proof. We follow the line of the proof of Theorem 4.1 in Zhao and Li (2002). Let (u_k, v_k, w_k, q_k) be defined as

$$(u_k, v_k, w_k, q_k) = \frac{1}{\rho_k} F_{\rho_k}(x_k, y_k, s_k, z_k) .$$

Then $\|(u_k, v_k, w_k, q_k)\|_\infty \leq \beta$ by the definition of $\mathcal{N}_\beta(\rho_k)$ and the following system of equations holds

$$\begin{aligned} &\operatorname{diag}(x_k)\, s_k = \rho_k (e + u_k) , \\ &\operatorname{diag}(y_k)\, z_k = \rho_k (e + v_k) , \\ &s_k = -A^T y_k + c + \kappa(\rho_k)^p B^T B (x_k - \hat{x}) + \rho_k w_k, \\ &z_k = A x_k - b + \kappa(\rho_k)^p y_k + \rho_k q_k. \end{aligned} \tag{36}$$

Let (x^*, y^*, s^*, z^*) be an optimal solution, i.e. a solution to equation (30). Then $(x^*, y^*, s^*, z^*) \geq 0$ and (s^*, z^*) is given by the expression

$$\begin{pmatrix} s^* \\ z^* \end{pmatrix} = \begin{pmatrix} 0 & -A^T \\ A & 0 \end{pmatrix} \begin{pmatrix} x^* \\ y^* \end{pmatrix} + \begin{pmatrix} c \\ -b \end{pmatrix} . \tag{37}$$

Due to (28) $(x^*)^T s^* = 0$ and $(y^*)^T z^* = 0$.

In what follows, we use the positive semidefiniteness

$$\begin{pmatrix} x \\ y \end{pmatrix}^T \begin{pmatrix} \kappa\rho^p B^T B & -A^T \\ A & \kappa\rho^p \mathbf{1} \end{pmatrix} \begin{pmatrix} x \\ y \end{pmatrix} = \kappa\rho^p \|Bx\|_2^2 + \kappa\rho^p \|y\|_2^2 . \tag{38}$$

Then we have

$$\begin{aligned}
0 \leq & \begin{pmatrix} x^* \\ y^* \end{pmatrix}^T \begin{pmatrix} s_k \\ z_k \end{pmatrix} + \begin{pmatrix} s^* \\ z^* \end{pmatrix}^T \begin{pmatrix} x_k \\ y_k \end{pmatrix} \\
= & \begin{pmatrix} x^* - x_k \\ y^* - y_k \end{pmatrix}^T \left(\begin{pmatrix} \kappa (\rho_k)^p B^T B & -A^T \\ A & \kappa (\rho_k)^p \mathbf{1} \end{pmatrix} \begin{pmatrix} x_k \\ y_k \end{pmatrix} + \begin{pmatrix} c \\ -b \end{pmatrix} \right. \\
& \left. + \rho_k \begin{pmatrix} w_k \\ q_k \end{pmatrix} - \begin{pmatrix} \kappa (\rho_k)^p B^T B \hat{x} \\ 0 \end{pmatrix} \right) + \begin{pmatrix} s^* \\ z^* \end{pmatrix}^T \begin{pmatrix} x_k \\ y_k \end{pmatrix} + \begin{pmatrix} x_k \\ y_k \end{pmatrix}^T \begin{pmatrix} s_k \\ z_k \end{pmatrix} \\
= & - \begin{pmatrix} x_k - x^* \\ y_k - y^* \end{pmatrix}^T \begin{pmatrix} \kappa (\rho_k)^p B^T B & -A^T \\ A & \kappa (\rho_k)^p \mathbf{1} \end{pmatrix} \begin{pmatrix} x_k - x^* \\ y_k - y^* \end{pmatrix} \\
& - \begin{pmatrix} x_k - x^* \\ y_k - y^* \end{pmatrix}^T \left(\begin{pmatrix} \kappa (\rho_k)^p B^T B & -A^T \\ A & \kappa (\rho_k)^p \mathbf{1} \end{pmatrix} \begin{pmatrix} x^* \\ y^* \end{pmatrix} + \begin{pmatrix} c \\ -b \end{pmatrix} \right. \\
& \left. + \rho_k \begin{pmatrix} w_k \\ q_k \end{pmatrix} - \begin{pmatrix} \kappa (\rho_k)^p B^T B \hat{x} \\ 0 \end{pmatrix} \right) + \begin{pmatrix} x_k \\ y_k \end{pmatrix}^T \begin{pmatrix} s^* \\ z^* \end{pmatrix} + \begin{pmatrix} x_k \\ y_k \end{pmatrix}^T \begin{pmatrix} s_k \\ z_k \end{pmatrix} \\
= & - \kappa (\rho_k)^p \left\| \begin{matrix} B (x_k - x^*) \\ y_k - y^* \end{matrix} \right\|_2^2 - \kappa (\rho_k)^p \begin{pmatrix} x_k - x^* \\ y_k - y^* \end{pmatrix}^T \begin{pmatrix} B^T B x^* + (\rho_k)^{1-p} \kappa^{-1} w_k - B^T B \hat{x} \\ y^* + (\rho_k)^{1-p} \kappa^{-1} q_k \end{pmatrix} \\
& + \begin{pmatrix} x_k \\ y_k \end{pmatrix}^T \begin{pmatrix} s_k \\ z_k \end{pmatrix}.
\end{aligned} \tag{39}$$

From equation (36) we obtain

$$(x_k)^T s_k = \rho_k e^T (e + u_k), \quad (y_k)^T z_k = \rho_k e^T (e + v_k)$$

and the estimate

$$\left| \begin{pmatrix} x_k \\ y_k \end{pmatrix}^T \begin{pmatrix} s_k \\ z_k \end{pmatrix} \right| \leq \rho_k c$$

for some positive constant c.

Finally we obtain

$$\left\| \begin{matrix} B (x_k - x^*) \\ y_k - y^* \end{matrix} \right\|_2^2 \leq \left\| \begin{matrix} x_k - x^* \\ y_k - y^* \end{matrix} \right\|_2 \left\| \begin{matrix} B^T B (x^* - \hat{x}) + (\rho_k)^{1-p} \kappa^{-1} w_k \\ y^* + (\rho_k)^{1-p} \kappa^{-1} q_k \end{matrix} \right\|_2 + (\rho_k)^{1-p} \kappa^{-1} c. \tag{40}$$

Since matrix B is invertible, we have

$$\|x\|_2 \leq \left\| B^{-1} \right\| \|Bx\|_2 .$$

Taking into account the above estimates and the estimate $\rho_k \leq \rho_0$, we obtain

$$\left\| \begin{matrix} x_k - x^* \\ y_k - y^* \end{matrix} \right\|_2 \leq c. \tag{41}$$

The boundedness of (s_k, z_k) follows from the above estimate and equation (36).

□

The complete algorithm is presented on Fig. 2. This algorithm requires an appropriate initial step: ρ_0 and a starting point $(x_0, y_0, s_0, z_0) \in \mathcal{N}_\beta(\rho_0)$. Hence we choose $(x_0, y_0, s_0, z_0) > 0$ such that $\|F_{\rho_0}(x_0, y_0, s_0, z_0)\|_\infty \leq \beta\rho_0$, which guarantees that $(x_0, y_0, s_0, z_0) \in \mathcal{N}_\beta(\rho_0)$.

The choice of (x_0, y_0, s_0, z_0) starts with $y_0 = e$ and $z_0 = \kappa\rho_0^p e$. We choose x_0 and s_0 so that the term $\kappa\rho_0^p B^T B(x_0 - \hat{x})$ is canceled. Taking $x_0 = \max(e, e + \hat{x})$ makes $x_0 > 0$. On the other hand, $\kappa\rho_0^p B^T B(x_0 - \hat{x})$ is a vector with nonzero components bounded from above by $\kappa\rho_0^p(1 + \|\hat{x}^-\|)$. Hence we define $s_{0,init} = B^T B(x_0 - \hat{x})$ and compute $\kappa = 1/\| \operatorname{diag}(x_0)s_{0,init}\|_\infty$. Then we take $s_0 = \kappa\rho_0^p s_{0,init}$. That procedure eliminates from $\|F_{\rho_0}\|_\infty$ the terms with ρ_0^p and replaces them by constants. We can now describe the construction of the initial point step-by-step:

1. Set $y_0 = e$.
2. Set $x_0 = \max(e, e + \hat{x})$.
3. Take $\rho_0 = \max \left(1, \left\| \begin{matrix} cA^T y_0 - c \\ -Ax_0 + b \end{matrix} \right\|_\infty \right) + \delta$

```
Input: Select positive constants: $p, \beta, b_1, b_2, \sigma \in (0,1)$. Select initial values $\rho_0$,
$\kappa \in (0,1]$ and $(x_0, y_0, s_0, z_0) > 0$ such that $(x_0, y_0, s_0, z_0) \in \mathcal{N}_\beta(\rho_0)$

Step 1
    If $F_{\rho_k}(x_k, y_k, s_k, z_k) = 0$
    set $(x_{k+1}, y_{k+1}, s_{k+1}, z_{k+1}) = (x_k, y_k, s_k, z_k)$ and go to Step 3
    Otherwise solve the linear system
```

$$\begin{pmatrix} \operatorname{diag}(s_k) + \kappa\rho_k^p \operatorname{diag}(x_k) B^T B & -\operatorname{diag}(x_k) A^T \\ \operatorname{diag}(y_k) A & \operatorname{diag}(z_k) + \kappa\rho_k^p \operatorname{diag}(y_k) \end{pmatrix} \begin{pmatrix} \Delta x \\ \Delta y \end{pmatrix}$$
$$= \begin{pmatrix} \rho_k e - \operatorname{diag}(x_k) \left(\kappa\rho_k^p \operatorname{diag}(x_k) B^T B (x_k - \hat{x}) - A^T y_k + c \right) \\ \rho_k e - \operatorname{diag}(y_k) \left(A x_k + \kappa\rho_k^p y_k - b \right) \end{pmatrix}$$

```
    and set
```

$$\begin{pmatrix} \Delta s \\ \Delta z \end{pmatrix} = \begin{pmatrix} \kappa\rho_k^p B^T B & -A^T \\ A & \kappa\rho_k^p \mathbf{1} \end{pmatrix} \begin{pmatrix} \Delta x \\ \Delta y \end{pmatrix} + \begin{pmatrix} \kappa\rho_k^p B^T B (x_k - \hat{x}) - A^T y_k - s_k + c \\ A x_k + \kappa\rho_k^p y_k - z_k - b \end{pmatrix}.$$

```
Step 2
    Find the step size $\lambda_k$ such that
    $(x_{k+1}, y_{k+1}, s_{k+1}, z_{k+1}) = (x_k + \lambda_k \Delta x, y_k + \lambda_k \Delta y, s_k + \lambda_k \Delta s, z_k + \lambda_k \Delta z) \in \mathcal{N}_\beta(\rho_k)$
    This is achieved in 2 substeps
    substep 1
    Find $\alpha$ such that $\forall \lambda \in (0, \alpha)$ $(x_k + \lambda \Delta x, y_k + \lambda \Delta y, s_k + \lambda \Delta s, z_k + \lambda \Delta z) > 0$
    substep 2
    Let $\lambda_k = \alpha b_1^j$, where $j$ is the smallest integer such that
```

$$\begin{aligned} &\left\| F_{\rho_k} (x_k + \lambda_k \Delta x, y_k + \lambda_k \Delta y, s_k + \lambda_k \Delta s, z_k + \lambda_k \Delta z) \right\|_\infty \\ &\quad \leq (1 - \sigma\lambda_k) \left\| F_{\rho_k} (x_k, y_k, s_k, z_k) \right\|_\infty . \end{aligned}$$

```
Step 3
    Find the smallest $\rho_{k+1} < \rho_k$ such that $(x_{k+1}, y_{k+1}, s_{k+1}, z_{k+1}) \in \mathcal{N}_\beta(\rho_{k+1})$.
    Let $\rho_{k+1} = (1 - b_2^j)\rho_k$ where $j$ is the smallest integer for which we have
```

$$\left\| F_{\rho_{k+1}} (x_{k+1}, y_{k+1}, s_{k+1}, z_{k+1}) \right\|_\infty \leq \beta \rho_{k+1}.$$

```
Stopping criterion: $\rho_{k+1} \leq tol$ or $\|F_{\rho_{k+1}}(x_{k+1}, y_{k+1}, s_{k+1}, z_{k+1})\|_\infty \leq tol$.
```

Figure 2: Regularized central path algorithm with projection

4. Set $s_{0,init} = B^T B(x_0 - \hat{x})$.
5. Compute $\kappa = 1/\|\operatorname{diag}(x_0)s_{0,init}\|_\infty$.
6. Set $z_0 = \kappa\rho_0^p e$.
7. Set $s_0 = \kappa\rho_0^p s_{0,init}$.
8. Compute
$$\xi = \frac{\|F_{\rho_0}(x_0, y_0, s_0, z_0)\|_\infty}{\rho_0}.$$
9. Set $\beta \in (\xi, 1)$.

Remark 2.4.3. Due to the above choice of the initial point we have the estimates

$$\begin{aligned}
&\|\operatorname{diag}(x_0)\, s_0 - \rho_0 e\|_\infty \le \left|\rho_0 - \rho_0^p\right| \le \rho_0,\\
&\|\operatorname{diag}(z_0)\, y_0 - \rho_0 e\|_\infty \le \kappa \left|\rho_0 - \rho_0^p\right| + (1-\kappa)\,\rho_0 \le \rho_0,\\
&\left\|s_0 + A^T y_0 - c - \kappa\rho_0^p B^T B\,(x_0 - \hat{x})\right\|_\infty \le \left\|A^T y_0 - c\right\|_\infty,\\
&\left\|z_0 - Ax_0 + b - \kappa\rho_0^p y_0\right\|_\infty \le \|-Ax_0 + b\|_\infty.
\end{aligned}$$

Hence
$$\left\|F_{\rho_0}(x_0, y_0, s_0, z_0)\right\|_\infty \le \rho_0$$
and for a properly chosen $\delta > 0$ the above inequality is sharp and $\xi < 1$. Then taking $\beta \in (\xi, 1)$ we obtain the desired estimate $\|F_{\rho_0}(x_0, y_0, s_0, z_0)\|_\infty \le \beta\rho_0$.

The computational complexity of our algorithm arises from the solution of the $(n+m)$-dimensional system (33). We can reduce the dimension of this system observing that it can be written as

$$\begin{aligned}
&\left(\operatorname{diag}(s_k) + \kappa\,(\rho_k)^p\, B^T B \operatorname{diag}(x_k)\right)\Delta x - \operatorname{diag}(x_k)\, A^T \Delta y\\
&\qquad = \rho_k e - \operatorname{diag}(x_k)\left(\kappa\,(\rho_k)^p\, B^T B\,(x_k - \hat{x}) - A^T y_k + c\right),\\
&\operatorname{diag}(y_k)\, A\Delta x + \left(\operatorname{diag}(z_k) + \kappa\,(\rho_k)^p \operatorname{diag}(y_k)\right)\Delta y\\
&\qquad = \rho_k e - \operatorname{diag}(y_k)\left(Ax_k + \kappa\,(\rho_k)^p\, y_k - b\right).
\end{aligned} \tag{42}$$

When $n \ge m$ eliminating Δx gives the equation

$$\begin{aligned}
H_k \Delta y = &\rho_k e - \operatorname{diag}(y_k)\left(Ax_k + \kappa\,(\rho_k)^p\, y_k - b\right) - \operatorname{diag}(y_k)\, A\\
&\times \left(\operatorname{diag}(s_k) + \kappa\,(\rho_k)^p \operatorname{diag}(x_k)\, B^T B\right)^{-1}\\
&\times \left(\rho_k e - \operatorname{diag}(x_k)\left(\kappa\,(\rho_k)^p\, B^T B\,(x_k - \hat{x}) - A^T y_k + c\right)\right),
\end{aligned} \tag{43}$$

where

$$\begin{aligned}
H_k = &\operatorname{diag}(z_k) + \kappa\,(\rho_k)^p \operatorname{diag}(y_k)\\
&+ \operatorname{diag}(y_k)\, A\left(\operatorname{diag}(s_k) + \kappa\,(\rho_k)^p \operatorname{diag}(x_k)\, B^T B\right)^{-1} \operatorname{diag}(x_k)\, A^T.
\end{aligned} \tag{44}$$

For Δx we obtain then

$$\begin{aligned}
\Delta x = &\left(\operatorname{diag}(s_k) + \kappa\,(\rho_k)^p \operatorname{diag}(x_k)\, B^T B\right)^{-1}\\
&\times \left(\operatorname{diag}(x_k)\, A^T \Delta y + \rho_k e - \operatorname{diag}(x_k)\left(\kappa\,(\rho_k)^p\, B^T B\,(x_k - \hat{x}) - A^T y_k + c\right)\right).
\end{aligned} \tag{45}$$

For $m > n$ we can eliminate Δy to obtain

$$\begin{aligned}
M_k \Delta x = &\rho_k e - \operatorname{diag}(x_k)\left(\kappa\,(\rho_k)^p\, B^T B\,(x_k - \hat{x}) - A^T y_k + c\right) + \operatorname{diag}(x_k)\, A^T\\
&\times \left(\operatorname{diag}(z_k) + \kappa\,(\rho_k)^p \operatorname{diag}(y_k)\right)^{-1}\left(\rho_k e - \operatorname{diag}(y_k)\left(Ax_k + \kappa\,(\rho_k)^p\, y_k - b\right)\right),
\end{aligned} \tag{46}$$

where

$$\begin{aligned}
M_k = &\operatorname{diag}(s_k) + \kappa\,(\rho_k)^p\, B^T B \operatorname{diag}(x_k)\\
&+ \operatorname{diag}(x_k)\, A^T\left(\operatorname{diag}(z_k) + \kappa\,(\rho_k)^p \operatorname{diag}(y_k)\right)^{-1} \operatorname{diag}(y_k)\, A.
\end{aligned} \tag{47}$$

For Δy we obtain then

$$\Delta y = \left(\operatorname{diag}(z_k) + \kappa(\rho_k)^p \operatorname{diag}(y_k)\right)^{-1} \\ \times \left(-\operatorname{diag}(y_k)\, A\Delta x + \rho_k e - \operatorname{diag}(y_k)\left(Ax_k + \kappa(\rho_k)^p y_k - b\right)\right). \tag{48}$$

Examples

We show now how the general projection algorithm described above can be used to find an optimal portfolio with the smallest distance to a given benchmark portfolio. We begin with the mean-CVaR optimization of Section LP computable portfolio problems. For simplicity, we take $\mathbb{X} = \{u\colon u \geq 0\}$. Then the linear program corresponding to this problem reads

$$\begin{cases} \xi + \frac{1}{1-\alpha}\sum_{n=1}^{N} p_n y_n \to \min, \\ y_n \geq 0, \quad n = 1,\dots,N, \\ y_n + \xi + r_n^T u \geq 0, \quad n = 1,\dots,N, \\ u^T\mu \geq r_0, \\ u \geq 0. \end{cases} \tag{49}$$

Given the benchmark portfolio w_b, we are looking for a solution of the above linear program with the additional constraint

$$\|u - w_b\|_2 \to \min. \tag{50}$$

A solution to this problem can be obtained by the central path projection algorithm. To this end, we define $x = (u, \xi, y)$, $c = (0_J, 1, p)$, where 0_J is the zero vector of length J and p is the N dimensional vector with entries p_n. Matrix A and vector b are given by the expressions

$$A = \begin{pmatrix} r & 0_N^T & \operatorname{diag}(e_N) \\ \mu & 0 & 0_N \end{pmatrix}, \quad b = \begin{pmatrix} c0_N^T \\ r_0 \end{pmatrix}, \tag{51}$$

where r is the $N \times J$ matrix of discrete returns, 0_N denotes the row vector of length N with zero entries and e_N – the row vector of length N with all entries equal 1. We take $\hat{x} = (w_b, 0, 0_N)$ and

$$B = \begin{pmatrix} \operatorname{diag}(e_J) & 0 \\ 0 & \operatorname{diag}(\varepsilon_{N+1}) \end{pmatrix}, \tag{52}$$

where ε_k is the row vector of length k with all entries equal ε.

With the above definitions, the solution obtained by the central path projection algorithm is an optimal solution to problem (49) with constraint (50).

For the mean-LSAD problem with $\mathbb{X} = \{u\colon u \geq 0\}$, the linear program is

$$\begin{cases} \sum_{n=1}^{N} p_n y_n \to \min, \\ y_n \geq 0, \quad n = 1,\dots,N, \\ y_n + \hat{r}_n^T u \geq 0, \quad n = 1,\dots,N, \\ u^T\mu \geq r_0, \\ u \geq 0. \end{cases} \tag{53}$$

To reduce this problem to the standard form appropriate for the central path projection algorithm, we define $x = (u, y)$, $c = (0_J, p)$, $\hat{x} = (w_b, 0_N)$ and matrices

$$A = \begin{pmatrix} \hat{r} & \operatorname{diag}(e_N) \\ \mu & 0_N \end{pmatrix}, \quad b = \begin{pmatrix} 0_N^T \\ r_0 \end{pmatrix}, \quad B = \begin{pmatrix} \operatorname{diag}(e_J) & 0 \\ 0 & \operatorname{diag}(\varepsilon_N) \end{pmatrix}, \tag{54}$$

where $\hat{r}$ is the $N \times J$ matrix of discrete centered returns. Running the central path projection algorithm with the above defined vectors and matrices, we obtain an optimal solution to problem (53) with constraint (50).

Numerical examples

The algorithms described in Sections Benders decomposition and Projection algorithm for portfolio optimization are implemented in R in the package **PortfolioOptim**. The package offers two public functions:

- `BDportfolio_optim` – which performs portfolio optimization using Benders decomposition;

- `PortfolioOptimProjection` – which implements the projection algorithm described in Section Projection algorithm for portfolio optimization.

For solving LP subproblems in `BDportfolio_optim`, we use function `Rglpk_solve_LP` from the R-package **Rglpk**. All computations are carried out on a computer with Intel Core i5-5200U processor with 8 GB RAM running Linux operating system.

Benders decomposition algorithm

It has been already observed by Künzi-Bay and Mayer (2006) that a good discrete approximation of a continuous distribution of returns requires large samples. The size of the sample depends on the goal of optimization procedure. It has been shown in Künzi-Bay and Mayer (2006) that the value of the objective function is accurately estimated with samples of size 10 000–20 000. Our experience shows that estimation of the optimal portfolio weights requires much larger samples.

Consider first the model used in Künzi-Bay and Mayer (2006): they provide the vector of mean μ and the covariance matrix Σ for monthly returns of 5 assets (MSCI.CH, MSCI.E, MSCI.W, Pictet.Bond and JPM.Global), and assume that the asset returns have joint normal distribution. We represent this distribution by a random sample of size N and solve the optimization problem (21) in which we impose additional constrains taking $\mathbb{X} = \{u\colon \sum_{k=1}^{K} u_k = 1, u \geq 0\}$. In all computations we take the target portfolio return $r_0 = 0.005$ and the CVaR confidence level $\alpha = 0.95$.

First we initialize computations with Künzi-Bay and Mayer (2006) data.

```
library(mvtnorm)
library(PortfolioOptim)

generate_data_normal <- function (means, covmat, num)
{
    k <- ncol(covmat)
    sim_data <- rmvnorm  (n=num, mean = means, sigma=covmat)
    sim_data <- matrix(num, k, data = sim_data)
    colnames(sim_data) <- colnames(covmat)
    prob <- matrix(1/num,num,1)
    mod_returns <- cbind(sim_data, prob)
    return (mod_returns)
}

prepare_data_KM <- function ()
{
sample_cov <- matrix(5,5, data = c( 0.003059 ,  0.002556 , 0.002327 , 0.000095 , 0.000533,
 0.002556 ,  0.003384 , 0.002929 ,  0.000032 ,  0.000762,
 0.002327 ,  0.002929 , 0.003509 ,  0.000036 ,  0.000908,
 0.000095 ,  0.000032 , 0.000036 ,  0.000069  , 0.000048 ,
 0.000533 ,  0.000762 , 0.000908  , 0.000048  , 0.000564))
sample_mean <- c( 0.007417,  0.005822,   0.004236,  0.004231,   0.005534)
colnames(sample_cov) <- c("MSCI.CH", "MSCI.E", "MSCI.W", "Pictet.Bond", "JPM.Global")
return(list( sample_mean = sample_mean, sample_cov = sample_cov))
}
```

We perform two tests (both tests are run simultaneously). In the first test, we compare computational time for different sample sizes. In the second test, we compare the accuracy of the obtained optimal portfolios. We perform computations for different sample sizes; for each sample size we generate 10 independent samples and assess the mean and the variance of the running time and portfolio weights. The R code for these tests is as follows.

```
data_nA <- prepare_data_KM()
sample_cov <- data_nA$sample_cov
sample_mean <- data_nA$sample_mean
k <- ncol(sample_cov)
a0 <- rep(1,k)
Aconstr <- rbind(a0,-a0)
bconstr <- c(1+1e-8, -1+1e-8)
lbound <- rep(0,k)
ubound <- rep(1,k)
R0 = 0.005
```

```
ET <- NULL
weights <- NULL
repetition = 10
sample_size = 10 000  # also run for 100 000 and 1 000 0000

ptm <- proc.time()[3]
  for (i in 1:repetition ){
    mod_returns <- generate_data_normal (sample_mean, sample_cov, sample_size)
    res  <-  BDportfolio_optim (mod_returns, R0, risk = "CVAR",  alpha = 0.95,
            Aconstr, bconstr,lbound, ubound, maxiter = 200, tol = 1e-10 )
    ET <- c(ET, proc.time()[3] - ptm)
    ptm <- proc.time()[3]
    weights <- rbind(weights, t(res$theta))
  }

cat(''running time and its standard deviation \n'')
print(mean(ET))
print(sqrt(var(ET)))

cat(''optimal portfolio and confidence intervals of its weights \n'')
print((colMeans(weights))*100)
print(sqrt(apply(weights, 2, var))*100*4/sqrt(repetition))
```

Since standard LP solvers (used for comparison by Künzi-Bay and Mayer (2006)) are too memory demanding to run on our hardware, we report computational time only for our BDportfolio_optim function. Table 1 contains the average running times for 10 different realizations together with the standard deviations.

sample size	mean	st. dev.
10 000	0.0663	0.0085
100 000	0.4953	0.0363
1 000 000	4.518	0.1086

Table 1: Averaged running time (in sec.) and its standard deviation for computations with different sample sizes.

The average running time for samples of size 10 000 is slightly smaller than the value reported by Künzi-Bay and Mayer (2006) (they give the value 0.088 sec.) but it can be attributed to a slightly faster CPU. It is also visible from Table 1 that even for samples of size 1 000 000 the computational time is small enough for such sample sizes to be used in practical computations.

The estimates of optimal portfolio weights for different sample sizes are collected in Table 2. The 95% confidence intervals of portfolio weights are reported in brackets. If the difference between portfolio weights is larger that two standard deviations we conclude that the difference is statistically significant (this corresponds to 95% two-sided Student-t test whose critical value for the sample of size 10 is 2.228). The results of Table 2 show that samples of size 10 000 are much to small to produce reliable portfolio weights. The values for that sample size are statistically significantly different from the values for 1 000 000 samples. From the values of confidence intervals of portfolio weights we can conclude that the computation of optimal portfolio weights from samples of size 10 000 can give values with an error up to 50% (for all non-zero weights and the sample of size 10 000 the confidence intervals are of order of one half of the corresponding weights). This is a clear indication that to obtain reliable values of portfolio weights we have to use samples of size 1 000 000 or take the average of a large number of repetitions, which is computationally equivalent.

We analyze now the effect of the number of assets in portfolio on the accuracy of computations. To this end, we use the data-set `etfdata` from the R-package **parma**. For comparison we take two subsets: `etfdata[1:500,1:5]`(with 5 assets) and `etfdata[1:500,1:10]` (with 10 assets). We add to our code a new function computing the vector of means and covariance matrix for a data set with k assets and perform computations with $k = 5$ and $k = 10$.

```
library(parma)
library(xts)
```

	sample size		
assets	10 000	100 000	1 000 000
MSCI.CH	9.6 (4.33)	11.2 (1.45)	10.9 (0.39)
MSCI.E	0.0 (0.00)	0.0 (0.00)	0.0 (0.00)
MSCI.W	0.0 (0.00)	0.0 (0.00)	0.0 (0.00)
Pictet.Bond	53.6 (13.24)	56.2 (2.33)	56.8 (0.83)
JPM.Global	36.8 (11.41)	32.6 (2.37)	32.3 (0.74)

Table 2: Optimal portfolios for different sample sizes. 95% confidence intervals of portfolio weights given in parentheses (all values are in percentage points).

```
library(quantmod)

data(etfdata)
prepare_data <- function (k)
{
    quotData <- as.xts(etfdata[1:500, 1:k])
    retData = NULL
    for (i in 1:k)
      retData <- cbind(retData,weeklyReturn(quotData[,i], type='arithmetic'))

    colnames(retData) <- colnames(quotData)
    sample_cov <- cov(retData)
    sample_mean <- colMeans(retData)
    return(list(sample_mean = sample_mean, sample_cov = sample_cov))
}

  data_nA <- prepare_data(5)  # also run with prepare_data(10)
  sample_cov <- data_nA$sample_cov
  sample_mean <- data_nA$sample_mean
  k <- ncol(sample_cov)
  a0 <- rep(1,k)
  Aconstr <- rbind(a0,-a0)
  bconstr <- c(1+1e-8, -1+1e-8)
  lbound <- rep(0,k)
  ubound <- rep(1,k)

  R0 = 0.004
  sample_size = 10 000  # also run for 100 000 and 1 000 0000
  repetition = 100
  weights <- NULL
  for (i in 1:repetition ){
      returns <- generate_data_normal(sample_mean, sample_cov, sample_size)
      res  <-  BDportfolio_optim (returns, R0, risk = "CVAR",  alpha = 0.95,
              Aconstr, bconstr, lbound, ubound, maxiter = 200, tol = 1e-10 )
      weights <- rbind(weights, t(res$theta))
  }
  print(sum(sqrt(apply(weights, 2, var))*100*4/sqrt(repetition))/k)
```

For each data set we generate samples of size 10 000, 100 000 and 1 000 000. For each sample size and data set we compute optimal portfolios. These computations are repeated 100 times and the empirical variances of portfolio weights are calculated. Using these results we compute widths of 95% confidence intervals of portfolio weights. To facilitate comparison of results between 5 and 10 assets we report in Table 3 the average width of 95% confidence interval per asset, i.e. the sum of widths for each asset divided by the number of assets. Notice that there is no significant difference in accuracy between portfolios of 5 and 10 assets. It can be interpreted as a kind of robustness of Benders decomposition algorithm. The results fit almost perfectly to the theoretical picture of the square root dependence of confidence intervals on sample size. The values in Table 3 for 1 000 000 samples confirm that with that sample size we can get almost perfect estimate of portfolio weights with the average confidence interval of 0.1–0.2% which is more than sufficient for practitioners.

	sample size		
assets number	10 000	100 000	1 000 000
5	0.97	0.29	0.09
10	1.18	0.42	0.15

Table 3: The average width of 95% confidence interval of portfolio weights (values are in percentage points).

Projection algorithm

In testing the projection algorithm described in Section Projection algorithm for portfolio optimization the following values of parameters are used $\delta = 0.5$, $p = 0.8$, $\sigma = 10^{-3}$, $b1 = 0.9$ and $b2 = 0.9$. The starting point in all computations is calculated by the procedure described in that Section. The algorithm is tested on simulated data generated from a normal distribution using the vectors of means and covariance matrices estimated from the previously used data sets `etfdata[1:500,1:5]` and `etfdata[1:500,1:10]` from the package **parma**. Samples of 125, 250 and 500 returns are generated assuming they are weekly returns (the vector of means and covariance matrix appropriately scaled) which corresponds to 2.5, 5 and 10 years of weekly data. For each sample we compute the vector of portfolio weights using the function PortfolioOptimProjection with tol = 1e-6 and taking as the benchmark the portfolio $1/J$, i.e. the portfolio with all weights equal to $1/J$, where J is the number of assets in the portfolio. The computation for each sample size and asset set is repeated 10 times.

The code is as follows

```
data_nA <- prepare_data(5) # also run with prepare_data(10)
sample_cov <- data_nA$sample_cov
sample_mean <- data_nA$sample_mean
k <- ncol(sample_cov)
a0 <- rep(1,k)
Aconstr <- rbind(a0,-a0)
bconstr <- c(1+1e-8, -1+1e-8)
lbound <- rep(0,k)
ubound <- rep(1,k)

w_m = rep(1/k,k)
R0 = 0.005
returns_size =125 # also run for 250 and 500
ET <- NULL
ptm <- proc.time()[3]
for (i in 1:10 ){
mod_returns <- generate_data_normal(sample_mean, sample_cov, returns_size )
res  <-  PortfolioOptimProjection (mod_returns, R0 , risk = "CVAR",  alpha = 0.95,
        w_m, Aconstr, bconstr,lbound, ubound, 800, tol = 1e-6 )
ET <- c(ET, proc.time()[3] - ptm)
ptm <- proc.time()[3]
}
cat(''Mean running time and its standard deviation \n'')
print(c( mean(ET), sqrt(var(ET))))
```

The average running time per one computation and its standard deviation is reported in Table 4.

Taking into account that the sample of size n corresponds to the matrix A of approximate dimension $n \times n$ the results of Table 4 are compatible with the results reported in Table 6.1 of Zhao and Li (2002). The observation which is missed in Zhao and Li (2002) is the standard deviation of the running time. This standard deviation is of the same order of magnitude as the average running time. We can conclude that the computational time depends not only on the size of matrix A but also on the entries,

number of assets	sample size	mean	sd. deviation
5	125	6.8618	5.7377
	250	35.1910	30.2499
	500	134.9312	95.2407
10	125	3.1451	1.6976
	250	30.0261	22.5226
	500	125.1935	104.7991

Table 4: Average running time and its standard deviation for the computation of optimal portfolios by the central path projection algorithm for different number of assets and different sample sizes (all values in sec. of CPU time).

which is particularly surprising since samples are drawn from the same distribution. Of course, the sample of size 125 can be considered as small even for returns of 5 assets. But the sample of size 500 is already quite large. In addition, the averaged running time is increasing with the sample size, however that increase is very irregular. At the same time, the standard deviation behaves also irregularly (similarly as for the averaged running time). All these observations show that the computation time is very sensitive not only to the dimensionality of the problem but also to a particular realization of the data. However we can still conclude that on average the algorithm is efficient for moderate-size portfolio optimization problems. Indeed, Zhao and Li (2002) already observed that the convergence rate of their algorithm was slow. The same is valid for our extension of their algorithm, which makes the algorithm not practically applicable for portfolio optimization problems when the size of the discrete distribution of returns is larger than 10^3. There is no clear indication which part of the algorithm has the main effect on slowing down the convergence.

Summary

We presented two optimization algorithms for financial portfolios. These algorithms find optimal portfolios in problems when the nonlinear optimization in mean-risk setting can be reformulated as a linear programming problem. The algorithm implementing Benders cuts allows for large samples. Our experience with very large data samples obtained by simulation from a given distribution shows that the algorithm is robust and estimated optimal portfolios are very stable (the standard deviation of portfolio weights is below 0.5 %). The second algorithm based on the central path projection can be useful for small data samples where solutions are not unique and form a multidimensional simplex. Extracting from this simplex a point with the smallest distance to a given benchmark portfolio can be used to improve the decision process in asset management. However, the second algorithm which implements the central path projection requires further analysis. In particular, we would like to make the algorithm's performance and running time less dependent on the realization of the data.

Acknowledgments

The author gratefully acknowledges financial support from National Science Centre, Poland, project 2014/13/B/HS4/00176.

Bibliography

J. F. Benders. Partitioning procedures for solving mixed-variables programming problems. *Numerische Mathematik*, 4:238–252, 1962. URL `https://doi.org/10.1007/BF01386316`. [p5]

A. Ghalanos. *parma: Portfolio Allocation and Risk Management Applications*, 2016. URL `https://cran.r-project.org/web/packages/parma/parma.pdf`. R package version 1.5-3. [p1]

W. K. Klein Haneveld and M. H. Van der Vlerk. Integrated chance constraints: Reduced forms and an algorithm. *Computational Management Science*, 3:245–269, 2006. URL `https://doi.org/10.1007/s10287-005-0007-3`. [p1, 5]

R. Koenker and I. Mizera. Convex optimization in R. *Journal of Statistical Software*, 60(5):1–23, 2014. URL `https://doi.org/10.18637/jss.v060.i05`. [p1]

H. Konno. Piecewise linear risk function and portfolio optimization. *Journal of the Operations Research Society of Japan*, 33:139–156, 1990. URL https://doi.org/10.15807/jorsj.33.139. [p3]

H. Konno and H. Yamazaki. Mean-absolute deviation portfolio optimization model and its application to Tokyo stock market. *Management Science*, 37:519–531, 1991. URL https://doi.org/10.1287/mnsc.37.5.519. [p3]

H. Konno, H. Waki, and A. Yuuki. Portfolio optimization under lower partial risk measures. *Asia-Pacific Financial Markets*, 9:127–140, 2002. URL https://doi.org/10.1023/A:1022238119491. [p3]

A. Künzi-Bay and J. Mayer. Computational aspects of minimizing conditional value at risk. *Computational Management Science*, 3:3–27, 2006. URL https://doi.org/10.1007/s10287-005-0042-0. [p1, 5, 15, 16]

R. Mansini, W. Ogryczak, and M. G. Speranza. Twenty years of linear programming based portfolio optimization. *European Journal of Operational Research*, 234:518–535, 2014. URL https://doi.org/10.1016/j.ejor.2013.08.035. [p1]

B. G. Peterson and P. Carl. *Portfolio Analysis, Including Numerical Methods for Optimization of Portfolios*, 2015. URL https://cran.r-project.org/web/packages/PortfolioAnalytics/PortfolioAnalytics.pdf. R package version 1.0.3636. [p1]

B. Pfaff. *Financial Risk Modelling and Portfolio Optimization with R, 2nd Ed.* John Wiley & Sons, 2016. ISBN 978-1-119-11966-1. [p1]

R. T. Rockafellar and S. Uryasev. Optimization of conditional value-at-risk. *Journal of Risk*, 2:21–42, 2000. URL https://doi.org/10.21314/JOR.2000.038. [p3]

R. T. Rockafellar, S. Uryasev, and M. Zabarankin. Master funds in portfolio analysis with general deviation measures. *Journal of Banking and Finance*, 30:743–778, 2006. URL https://doi.org/10.1016/j.jbankfin.2005.04.004. [p2]

S. Theussl and H. W. Borchers. *CRAN Task View: Optimization and Mathematical Programming*. CRAN, 2016. URL http://cran.r-project.org/web/views/Optimization.html. [p1]

D. Würtz, Y. Chalabi, W. Chen, and A. E. Pfaff. *Portfolio Optimization with R/Rmetrics*. Rmetrics Association & Finance Online, 2009. [p1]

Y.-B. Zhao and D. Li. Locating the least 2-norm solution of linear programs via a path-following method. *SIAM Journal on Optimization*, 12(4):893–912, 2002. URL https://doi.org/10.1137/S1052623401386368. [p1, 2, 8, 10, 18, 19]

Andrzej Palczewski
Faculty of Mathematics, Informatics and Mechanics, University of Warsaw
Banacha 2, 02-097 Warsaw
Poland
A.Palczewski@mimuw.edu.pl