Comput Optim Appl (2012) 52:645–666
DOI 10.1007/s10589-011-9436-9

A polynomial optimization approach to constant rebalanced portfolio selection

Yuichi Takano · Renata Sotirov

Received: 26 October 2010 / Published online: 21 September 2011

Abstract We address the multi-period portfolio optimization problem with the constant rebalancing strategy. This problem is formulated as a polynomial optimization problem (POP) by using a mean-variance criterion. In order to solve the POPs of high degree, we develop a cutting-plane algorithm based on semidefinite programming. Our algorithm can solve problems that can not be handled by any of known polynomial optimization solvers.

Keywords Multi-period portfolio optimization · Polynomial optimization problem · Constant rebalancing · Semidefinite programming · Mean-variance criterion

1 Introduction

We consider the constant rebalancing strategy (also referred to as constant mix, fixed mix, constant proportional portfolio and the like) in the multi-period portfolio selection. In this strategy, we rebalance the portfolio at the beginning of every period so that the investment proportion will be restored to the fixed constant one. This strategy is widely used in business. Moreover, it is known that constant rebalancing achieves the optimal growth rate of wealth if the asset prices in each period are independent and identically distributed (i.i.d.) (see e.g., [1]). On the assumptions of i.i.d. and infinite horizon, the problem to be solved is a relatively easy convex program (e.g.,

Y. Takano
Department of Industrial Engineering and Management, Graduate School of Decision Science and Technology, Tokyo Institute of Technology, 2-12-1-W9-77 Ookayama, Meguro-ku, Tokyo 152-8552, Japan
e-mail: takano.y.ad@m.titech.ac.jp

R. Sotirov (✉)
Department of Econometrics and Operations Research, Tilburg University, Warandelaan 2, 5000 LE Tilburg, The Netherlands
e-mail: r.sotirov@uvt.nl

[3, 17]). However, the constant rebalancing strategy generally leads to nonconvex optimization. Because of its difficulty, most studies (e.g., [5, 26]) have focused on approximately solving the constant rebalanced portfolio optimization problem. To the best of our knowledge, only Maranas et al. [19] approached it through global optimization by developing a specialized branch-and-bound algorithm.

In this paper, we use a mean-variance (M-V) criterion to formulate the constant rebalanced portfolio optimization problem (see also [19]) as a polynomial optimization problem (POP). Although solving POPs has been a challenging task for decades, recently it turned out that small and medium size POPs can be efficiently solved by using semidefinite programming (SDP) [10, 11, 15, 22]. In 2001, Lasserre [15] introduced a hierarchy of SDP relaxations and proved that the sequence of obtained lower bounds monotonically converges to the global optimum of the corresponding POP. However, it is difficult to solve the corresponding large-scale SDP relaxations when a POP contains many decision variables and/or a polynomial of high degree. Although some specific problem structures can be exploited to reduce the size of SDP relaxations [6, 9, 12, 16, 27], our problem does not have any such special structure. As a result, when the number of planning periods is large, our POP is intractable for mentioned approaches due to monomials of high degree. In order to solve POPs of high degree, we develop a cutting-plane algorithm that solves in each iteration a POP of reduced degree and converges to an optimum of the original POP.

We conduct computational experiments, and assess the benefit of our cutting plane approach in comparison with the global optimization solver over polynomials GloptiPoly [7], the global optimization solver BARON [24], and the nonlinear programming (NLP) solver CONOPT [4]. GloptiPoly, which builds up a hierarchy of SDP relaxations (see [15]), successfully provides a globally optimal solution of small-size problems. Solutions obtained by CONOPT do not have a guarantee of global optimality whereas BARON seeks a globally optimal solution. By using our cutting-plane algorithm we solve problems, which were too large to be directly solved by GloptiPoly, faster than BARON. However, CONOPT reached a locally optimal solution of these problems in very short time.

The rest of the paper is organized as follows. In Sect. 2, we present the constant rebalancing strategy and formulate the M-V portfolio optimization problem. In Sect. 3, our cutting-plane algorithm is established and its application to the M-V portfolio optimization is described. Here we also prove a global convergence of the algorithm. The results of computational experiments are given in Sect. 4. Concluding remarks are given in Sect. 5.

2 Mean-variance portfolio optimization with constant rebalancing strategy

In this section, we provide a mathematical description of a portfolio dynamics under the constant rebalancing strategy (see also [19]). Further, we derive two equivalent formulations of the M-V portfolio optimization with the constant rebalancing strategy.

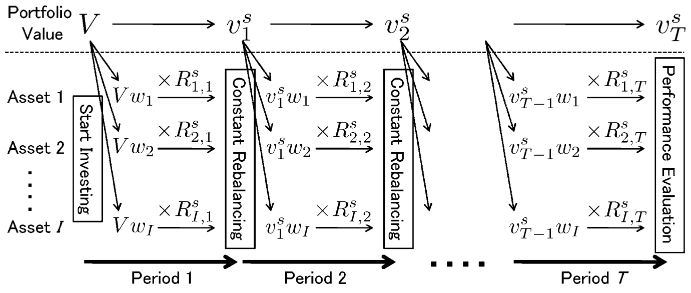


Fig. 1 Portfolio dynamics under scenario s

2.1 Constant rebalancing strategy and portfolio dynamics

We define the terminology and notation as follows:

$\mathbb{R}^N$: set of N-dimensional real vectors
$\mathbb{Z}_+^N$: set of N-dimensional nonnegative integer vectors

Index sets

$\mathcal{I} := \{1, 2, \ldots, I\}$: index set of investable financial assets
$\mathcal{T} := \{1, 2, \ldots, T\}$: index set of planning periods
$\mathcal{S} := \{1, 2, \ldots, S\}$: index set of given scenarios

Decision variables

v_t^s: portfolio value at the end of period t under scenario s $(t \in \mathcal{T}, s \in \mathcal{S})$
w_i: investment proportion in asset i $(i \in \mathcal{I})$ (where $\boldsymbol{w} := (w_1, w_2, \ldots, w_I) \in \mathbb{R}^I$)

Given constants

V: initial wealth for investment
$R_{i,t}^s$: total return of asset i at period t under scenario s $(i \in \mathcal{I}, t \in \mathcal{T}, s \in \mathcal{S})$
P_s: occurrence probability of scenario s $(s \in \mathcal{S})$
L_i (U_i): lower (upper) bound of the investment proportion in asset i $(i \in I)$

User-defined parameters

λ: trade-off parameter between return and risk (where $\lambda \in (0, 1)$)

We consider a self-financing portfolio and assume that there are no transaction costs. Note that

$$\sum_{s \in \mathcal{S}} P_s = 1 \quad \text{and} \quad P_s > 0 \quad \text{for all } s \in \mathcal{S}. \tag{1}$$

Figure 1 illustrates a portfolio dynamics under scenario s. Suppose that the amount of money V is provided for investment, and that one starts investing Vw_i in each asset i at the beginning of the planning horizon. Because of the return of each asset, the invested amount Vw_i is changed to $R_{i,1}^s Vw_i$ over the first period. Accordingly, the portfolio value at the end of the first period under scenario s is given by

$$v_1^s = \sum_{i \in \mathcal{I}} R_{i,1}^s V w_i = V \sum_{i \in \mathcal{I}} R_{i,1}^s w_i. \tag{2}$$

The constant rebalancing strategy enforces the rebalancing to the proportion $\boldsymbol{w}$ at the beginning of each investment period. The amount $R_{i,1}^s V w_i$ is adjusted to $v_1^s w_i$ at the beginning of the second period. Because of the return of each asset, the invested amount $v_1^s w_i$ is changed to $R_{i,2}^s v_1^s w_i$ over the second period. Accordingly, the portfolio value at the end of the second period under scenario s is given by

$$v_2^s = \sum_{i \in \mathcal{I}} R_{i,2}^s v_1^s w_i = v_1^s \sum_{i \in \mathcal{I}} R_{i,2}^s w_i. \tag{3}$$

Similarly to the first and second period, the investment proportion is adjusted according to the constant rebalancing strategy, and the portfolio value changes due to the return of each asset. Thus, the portfolio value at the end of the planning horizon of T periods under scenario s is given by

$$\begin{aligned} v_T^s &= v_{T-1}^s \sum_{i \in \mathcal{I}} R_{i,T}^s w_i \\ &= v_{T-2}^s \left(\sum_{i \in \mathcal{I}} R_{i,T-1}^s w_i \right) \left(\sum_{i \in \mathcal{I}} R_{i,T}^s w_i \right) = \cdots = V \prod_{t \in \mathcal{T}} \left(\sum_{i \in \mathcal{I}} R_{i,t}^s w_i \right). \end{aligned} \tag{4}$$

2.2 Two formulations of constant rebalanced portfolio optimization

We consider the following two performance measures:

Mean of the portfolio value:

$$\sum_{s \in \mathcal{S}} P_s v_T^s \overset{(4)}{=} \sum_{s \in \mathcal{S}} P_s V \prod_{t \in \mathcal{T}} \left(\sum_{i \in \mathcal{I}} R_{i,t}^s w_i \right) \tag{5}$$

Variance of the portfolio value:

$$\begin{aligned} \operatorname{Var}(\boldsymbol{w}) &:= \sum_{s \in \mathcal{S}} P_s (v_T^s)^2 - \left(\sum_{s \in \mathcal{S}} P_s v_T^s \right)^2 \\ &\overset{(4)}{=} \sum_{s \in \mathcal{S}} P_s \left(V \prod_{t \in \mathcal{T}} \left(\sum_{i \in \mathcal{I}} R_{i,t}^s w_i \right) \right)^2 - \left(\sum_{s \in \mathcal{S}} P_s V \prod_{t \in \mathcal{T}} \left(\sum_{i \in \mathcal{I}} R_{i,t}^s w_i \right) \right)^2 \end{aligned} \tag{6}$$

The former is the return measure and the latter is the risk measure. A framework of the M-V optimization for single-period portfolio selection was initially constructed by Markowitz [20], and this framework was also implemented in the multi-period portfolio selection, see [19].

We consider both, minimizing the variance of the portfolio value and maximizing the mean of the portfolio value, at the same time by taking the weighted sum of them. The constant rebalanced portfolio optimization problem is reduced to the following problem with I decision variables and simple linear constraints (see [19]):

$$\begin{aligned} \underset{\boldsymbol{w}\in\mathbb{R}^I}{\text{minimize}} \quad & (1-\lambda)\left(\sum_{s\in\mathcal{S}} P_s \left(V\prod_{t\in\mathcal{T}}\left(\sum_{i\in\mathcal{I}} R^s_{i,t} w_i\right)\right)^2 \right.\\ & \left. - \left(\sum_{s\in\mathcal{S}} P_s V\prod_{t\in\mathcal{T}}\left(\sum_{i\in\mathcal{I}} R^s_{i,t} w_i\right)\right)^2\right) \\ & - \lambda\left(\sum_{s\in\mathcal{S}} P_s V\prod_{t\in\mathcal{T}}\left(\sum_{i\in\mathcal{I}} R^s_{i,t} w_i\right)\right) \\ \text{subject to} \quad & \sum_{i\in\mathcal{I}} w_i = 1; \quad L_i \le w_i \le U_i, \quad i\in\mathcal{I}. \end{aligned} \tag{7}$$

We refer to (7) as the *NLP formulation*.

In the sequel, we reformulate the optimization problem (7) as a POP by using the following notation:

$$\boldsymbol{\alpha} := (\alpha_1, \alpha_2, \ldots, \alpha_I) \in \mathbb{Z}^I_+ \quad \text{and} \quad \boldsymbol{w}^{\boldsymbol{\alpha}} := \prod_{i\in\mathcal{I}} w_i^{\alpha_i}.$$

Following the above notation, the mean and variance of the portfolio value are transformed into polynomial forms as follows:

$$\begin{aligned} & \sum_{s\in\mathcal{S}} P_s V \prod_{t\in\mathcal{T}}\left(\sum_{i\in I} R^s_{i,t} w_i\right) \\ & = \sum_{s\in\mathcal{S}} P_s \sum_{\boldsymbol{\alpha}:\,\sum_{i\in\mathcal{I}}\alpha_i = T} \underbrace{\left(V \sum_{\substack{(i_1,i_2,\ldots,i_T):\\ |\{t\in\mathcal{T}\,|\,i_t=i\}|=\alpha_i,\, i\in\mathcal{I}}} \prod_{t\in\mathcal{T}} R^s_{i_t,t}\right)}_{\mathcal{C}_1(\boldsymbol{\alpha},s)} \boldsymbol{w}^{\boldsymbol{\alpha}} \\ & = \sum_{\boldsymbol{\alpha}:\,\sum_{i\in\mathcal{I}}\alpha_i = T} \underbrace{\left(\sum_{s\in\mathcal{S}} P_s \mathcal{C}_1(\boldsymbol{\alpha}, s)\right)}_{\mathcal{C}_2(\boldsymbol{\alpha})} \boldsymbol{w}^{\boldsymbol{\alpha}}, \end{aligned} \tag{8}$$

and

$$\sum_{s\in\mathcal{S}} P_s \left(V\prod_{t\in\mathcal{T}}\left(\sum_{i\in\mathcal{I}} R^s_{i,t} w_i\right)\right)^2 - \left(\sum_{s\in\mathcal{S}} P_s V\prod_{t\in\mathcal{T}}\left(\sum_{i\in I} R^s_{i,t} w_i\right)\right)^2$$

$$
\overset{(8)}{=} \sum_{s \in \mathcal{S}} P_s \left(\sum_{\substack{\boldsymbol{\alpha} :\, \sum\limits_{i \in \mathcal{I}} \alpha_i = T}} \mathcal{C}_1(\boldsymbol{\alpha}, s) \boldsymbol{w}^{\boldsymbol{\alpha}} \right)^2 - \left(\sum_{\substack{\boldsymbol{\alpha} :\, \sum\limits_{i \in \mathcal{I}} \alpha_i = T}} \mathcal{C}_2(\boldsymbol{\alpha}) \boldsymbol{w}^{\boldsymbol{\alpha}} \right)^2
$$

$$
= \sum_{\substack{\boldsymbol{\alpha} :\, \sum\limits_{i \in \mathcal{I}} \alpha_i = 2T}} \underbrace{\left(\sum_{\substack{\boldsymbol{\alpha}' : \boldsymbol{\alpha}' \le \boldsymbol{\alpha}, \\ \sum\limits_{i \in \mathcal{I}} \alpha'_i = T}} \sum_{s \in \mathcal{S}} P_s \mathcal{C}_1(\boldsymbol{\alpha}', s) \mathcal{C}_1(\boldsymbol{\alpha} - \boldsymbol{\alpha}', s) \right)}_{\mathcal{C}_3(\boldsymbol{\alpha})} \boldsymbol{w}^{\boldsymbol{\alpha}}
$$

$$
- \sum_{\substack{\boldsymbol{\alpha} :\, \sum\limits_{i \in \mathcal{I}} \alpha_i = 2T}} \underbrace{\left(\sum_{\substack{\boldsymbol{\alpha}' : \boldsymbol{\alpha}' \le \boldsymbol{\alpha}, \\ \sum\limits_{i \in \mathcal{I}} \alpha'_i = T}} \mathcal{C}_2(\boldsymbol{\alpha}') \mathcal{C}_2(\boldsymbol{\alpha} - \boldsymbol{\alpha}') \right)}_{\mathcal{C}_4(\boldsymbol{\alpha})} \boldsymbol{w}^{\boldsymbol{\alpha}}.
$$

Now, (7) is reformulated as the following POP:

$$
\begin{aligned}
& \underset{\boldsymbol{w} \in \mathbb{R}^I}{\text{minimize}} && \mathrm{OF}(\boldsymbol{w}) := (1 - \lambda) \sum_{\substack{\boldsymbol{\alpha} :\, \sum\limits_{i \in \mathcal{I}} \alpha_i = 2T}} \mathcal{C}_5(\boldsymbol{\alpha}) \boldsymbol{w}^{\boldsymbol{\alpha}} - \lambda \sum_{\substack{\boldsymbol{\alpha} :\, \sum\limits_{i \in \mathcal{I}} \alpha_i = T}} \mathcal{C}_2(\boldsymbol{\alpha}) \boldsymbol{w}^{\boldsymbol{\alpha}} \\
& \text{subject to} && \sum_{i \in \mathcal{I}} w_i = 1; \quad L_i \le w_i \le U_i, \quad i \in \mathcal{I},
\end{aligned} \tag{9}
$$

where $\mathcal{C}_5(\boldsymbol{\alpha}) := \mathcal{C}_3(\boldsymbol{\alpha}) - \mathcal{C}_4(\boldsymbol{\alpha})$. Note that the degree of monomials in $\mathrm{OF}(\boldsymbol{w})$ are T and $2T$. In the sequel, we refer to (9) as the *POP formulation.*

Remark 1 By using $w_I = 1 - \sum_{i=1}^{I-1} w_i$, we can eliminate w_I in (9). However, we do not implement this elimination because we do not expect significant impact on computations.

3 Cutting-plane algorithm

If there is a polynomial of high degree in a POP, then the relaxation order, ω (for details see [15]) is also high. Accordingly, the corresponding SDP relaxations are also large-scale, and consequently, it is hard to solve them. In this section, we develop a cutting-plane algorithm to approximately solve POPs of higher degree. The main idea of the algorithm is to exploit the structure of the problem to obtain tractable subproblems that iteratively converge to the original problem.

3.1 General form of the algorithm

Let us consider the following optimization problem:

$$\begin{aligned} \underset{\boldsymbol{x}\in\mathbb{R}^N}{\text{minimize}} \quad & r_0(\boldsymbol{x}) \\ \text{subject to} \quad & g_j(\boldsymbol{h}^j(\boldsymbol{x})) + r_j(\boldsymbol{x}) \leq 0, \quad j \in \mathcal{J}_1 := \{1, 2, \dots, J_1\} \\ & r_j(\boldsymbol{x}) \leq 0, \quad j \in \mathcal{J}_2 := \{J_1+1, J_1+2, \dots, J_2\}, \end{aligned} \tag{10}$$

where $g_j : \mathbb{R}^{N'} \to \mathbb{R}$, $j \in \mathcal{J}_1$ are continuously differentiable convex functions, and $\boldsymbol{h}^j = (h_1^j, h_2^j, \dots, h_{N'}^j)^\top : \mathbb{R}^N \to \mathbb{R}^{N'}$, $j \in \mathcal{J}_1$ and $r_j : \mathbb{R}^N \to \mathbb{R}$, $j = 0, 1, \dots, J_2$ are polynomial functions. Note that the functions g_j do not need to be polynomials. A problem of minimizing $g_0(\boldsymbol{h}^0(\boldsymbol{x})) + r_0(\boldsymbol{x})$, where g_0 is a continuously differentiable convex function and $\boldsymbol{h}^0$ is a polynomial function, can be easily converted to the form (10) by rewriting the problem as

$$\begin{aligned} \underset{(\boldsymbol{x},z)\in\mathbb{R}^N\times\mathbb{R}}{\text{minimize}} \quad & z + r_0(\boldsymbol{x}) \\ \text{subject to} \quad & g_0(\boldsymbol{h}^0(\boldsymbol{x})) - z \leq 0; \qquad g_j(\boldsymbol{h}^j(\boldsymbol{x})) + r_j(\boldsymbol{x}) \leq 0, \quad j \in \mathcal{J}_1 \\ & r_j(\boldsymbol{x}) \leq 0, \quad j \in \mathcal{J}_2. \end{aligned} \tag{11}$$

Let us assume that $\bar{\boldsymbol{x}}$ is an element of the following set:

$$\left\{\boldsymbol{x} \in \mathbb{R}^N \mid r_j(\boldsymbol{x}) \leq 0,\ j \in \mathcal{J}_2\right\}. \tag{12}$$

From the convexity of g_j, $j \in \mathcal{J}_1$, we have the following inequalities:

$$\begin{aligned} & g_j(\boldsymbol{h}^j(\boldsymbol{x}^2)) \geq g_j(\boldsymbol{h}^j(\boldsymbol{x}^1)) + \nabla g_j(\boldsymbol{h}^j(\boldsymbol{x}^1))^\top \left(\boldsymbol{h}^j(\boldsymbol{x}^2) - \boldsymbol{h}^j(\boldsymbol{x}^1)\right) \\ & \quad \text{for all } \boldsymbol{x}^1, \boldsymbol{x}^2 \in \mathbb{R}^N, \end{aligned} \tag{13}$$

where ∇g_j is the gradient of $g_j(\boldsymbol{x}')$ with respect to $\boldsymbol{x}' \in \mathbb{R}^{N'}$. Now, it follows from (13) that for each $\bar{\boldsymbol{x}}$ from (12) we have

$$\begin{aligned} & g_j(\boldsymbol{h}^j(\boldsymbol{x})) + r_j(\boldsymbol{x}) \\ & \quad \geq g_j(\boldsymbol{h}^j(\bar{\boldsymbol{x}})) + r_j(\boldsymbol{x}) + \nabla g_j(\boldsymbol{h}^j(\bar{\boldsymbol{x}}))^\top \left(\boldsymbol{h}^j(\boldsymbol{x}) - \boldsymbol{h}^j(\bar{\boldsymbol{x}})\right) \quad \text{for all } \boldsymbol{x} \in \mathbb{R}^N. \end{aligned}$$

This implies that every feasible point, $\boldsymbol{x}$, of the problem (10) satisfies the following constraint:

$$g_j(\boldsymbol{h}^j(\bar{\boldsymbol{x}})) + r_j(\boldsymbol{x}) + \nabla g_j(\boldsymbol{h}^j(\bar{\boldsymbol{x}}))^\top \left(\boldsymbol{h}^j(\boldsymbol{x}) - \boldsymbol{h}^j(\bar{\boldsymbol{x}})\right) \leq 0, \tag{14}$$

for given $\bar{\boldsymbol{x}}$. Moreover, if $\bar{\boldsymbol{x}}$ from (12) is an infeasible point of the problem (10), that is, $g_j(\boldsymbol{h}^j(\bar{\boldsymbol{x}})) + r_j(\bar{\boldsymbol{x}}) > 0$ for some $j \in \mathcal{J}_1$, then it is clear that $\boldsymbol{x} := \bar{\boldsymbol{x}}$ does not satisfy the constraint (14). Therefore, we can use (14) to separate an infeasible point $\bar{\boldsymbol{x}}$ from the feasible region. Note that (14) is a polynomial inequality in $\boldsymbol{x}$.

The fundamental principle of our algorithm, which is regarded as a natural extension of Kelley's convex cutting-plane algorithm (see e.g., Sect. 14.8 of [18]), is to solve a sequence of relaxed POPs and to approximate the feasible region of the original problem by cutting off the current infeasible solution of the relaxed problem.

A general form of the algorithm is as follows:

Algorithm GCP: General Form of Cutting-Plane Algorithm for Solving Problem (10)

Step 0. (Initialization) Define $\mathcal{X}$ as (12). Set $k \leftarrow 1$.

Step 1. (Lower Bound Estimation) Solve the following POP (for instance, by using a hierarchy of SDP relaxations of [15]):

$$\underset{\boldsymbol{x}\in\mathbb{R}^N}{\text{minimize}} \quad r_0(\boldsymbol{x}) \quad \text{subject to } \boldsymbol{x} \in \mathcal{X}. \tag{15}$$

Let $\bar{\boldsymbol{x}}^k$ be the solution.

Step 2. (Feasibility Check) Let $\mathcal{J}^k := \{j \in \mathcal{J}_1 \mid g_j(\boldsymbol{h}^j(\bar{\boldsymbol{x}}^k)) + r_j(\bar{\boldsymbol{x}}^k) > 0\}$. If $\mathcal{J}^k = \emptyset$, terminate the algorithm with the solution $\bar{\boldsymbol{x}}^k$.

Step 3. (Cut Generation) Set
$\mathcal{X} \leftarrow \mathcal{X} \cap \{\boldsymbol{x} \mid g_j(\boldsymbol{h}^j(\bar{\boldsymbol{x}}^k)) + r_j(\boldsymbol{x}) + \nabla g_j(\boldsymbol{h}^j(\bar{\boldsymbol{x}}^k))^\top \left(\boldsymbol{h}^j(\boldsymbol{x}) - \boldsymbol{h}^j(\bar{\boldsymbol{x}}^k)\right) \leq 0,\ j \in \mathcal{J}^k\}$,
and $k \leftarrow k+1$. Return to Step 1.

Let us suppose that the functions g_j are convex polynomials. Then, in order to apply the SDP approach of [15] to the problem (10), the relaxation order is

$$\omega \geq \max_{j\in\mathcal{J}_1} \lceil \deg(g_j(\boldsymbol{h}^j))/2 \rceil, \quad \text{and} \quad \omega \geq \max_{j=0,1,\dots,J_2} \lceil \deg(r_j)/2 \rceil$$

from its definition (see [15]), where $\deg(f)$ is the degree of the polynomial f. To the contrary, when we use the SDP approach of [15] in the above algorithm, ω has to satisfy the following conditions:

$$\omega \geq \max_{\substack{j\in\mathcal{J}_1 \\ \ell=1,2,\dots,N'}} \lceil \deg(h^j_\ell)/2 \rceil, \quad \text{and} \quad \omega \geq \max_{j=0,1,\dots,J_2} \lceil \deg(r_j)/2 \rceil.$$

We can prove convergence to the global optimum similarly to Kelley's convex cutting-plane algorithm under strong assumptions.

Theorem 1 *Suppose that the set* (12) *is compact, and that we always obtain a globally optimal solution of the problem* (15). *Then, every accumulation point of the sequence of solutions* $\{\bar{\boldsymbol{x}}^k\}$ *generated by Algorithm GCP is a globally optimal solution of the problem* (10).

Proof Suppose that $\mathcal{K} \subseteq \mathbb{Z}_+$, and the sequence $\{\bar{\boldsymbol{x}}^k\}_{k\in\mathcal{K}}$ converges to an accumulation point $\bar{\boldsymbol{x}}$. Then, for $k' > k$ $(k, k' \in \mathcal{K})$ it follows from Step 3 of GCP algorithm:

$$g_j(\boldsymbol{h}^j(\bar{\boldsymbol{x}}^k)) + r_j(\bar{\boldsymbol{x}}^{k'}) + \nabla g_j(\boldsymbol{h}^j(\bar{\boldsymbol{x}}^k))^\top \left(\boldsymbol{h}^j(\bar{\boldsymbol{x}}^{k'}) - \boldsymbol{h}^j(\bar{\boldsymbol{x}}^k)\right) \leq 0, \quad j \in \mathcal{J}^k,$$

and from (13) and the definition of $\mathcal{J}^k$:

$$g_j(\boldsymbol{h}^j(\bar{\boldsymbol{x}}^{k'})) + r_j(\bar{\boldsymbol{x}}^k) + \nabla g_j(\boldsymbol{h}^j(\bar{\boldsymbol{x}}^{k'}))^\top \left(\boldsymbol{h}^j(\bar{\boldsymbol{x}}^k) - \boldsymbol{h}^j(\bar{\boldsymbol{x}}^{k'})\right)$$
$$\overset{(13)}{\leq} g_j(\boldsymbol{h}^j(\bar{\boldsymbol{x}}^k)) + r_j(\bar{\boldsymbol{x}}^k) \leq 0, \quad j \notin \mathcal{J}^k.$$

By means of the Cauchy-Schwarz inequality, we have, for all $j \in \mathcal{J}_1$,

$$\begin{aligned} &\min\left\{ g_j(\boldsymbol{h}^j(\bar{\boldsymbol{x}}^k)) + r_j(\bar{\boldsymbol{x}}^{k'}),\, g_j(\boldsymbol{h}^j(\bar{\boldsymbol{x}}^{k'})) + r_j(\bar{\boldsymbol{x}}^k) \right\} \\ &\leq \max\left\{ \left|\nabla g_j(\boldsymbol{h}^j(\bar{\boldsymbol{x}}^k))\right|^\top \left|\left(\boldsymbol{h}^j(\bar{\boldsymbol{x}}^{k'}) - \boldsymbol{h}^j(\bar{\boldsymbol{x}}^k)\right)\right|, \right. \\ &\qquad \left. \left|\nabla g_j(\boldsymbol{h}^j(\bar{\boldsymbol{x}}^{k'}))\right|^\top \left|\left(\boldsymbol{h}^j(\bar{\boldsymbol{x}}^k) - \boldsymbol{h}^j(\bar{\boldsymbol{x}}^{k'})\right)\right| \right\}. \end{aligned} \tag{16}$$

Since $\mathcal{X}$ is compact and g_j is continuously differentiable, $|\nabla g_j(\boldsymbol{h}^j(\bar{\boldsymbol{x}}^k))|$ and $|\nabla g_j(\boldsymbol{h}^j(\bar{\boldsymbol{x}}^{k'}))|$ are bounded. So, the right-hand side of (16) goes to zero as k and k' go to infinity. The left-hand side of (16) goes to $g_j(\boldsymbol{h}^j(\bar{\boldsymbol{x}})) + r_j(\bar{\boldsymbol{x}})$ for the accumulation point $\bar{\boldsymbol{x}}$. Therefore, $\bar{\boldsymbol{x}}$ is a feasible solution of the problem (10).

Let r^* be the global optimum of the problem (10). Then, it follows that $r_0(\bar{\boldsymbol{x}}^k) \leq r^*$ for each $k \in \mathcal{K}$, and hence $r_0(\bar{\boldsymbol{x}}) \leq r^*$. Since $\bar{\boldsymbol{x}}$ is a feasible solution of the problem (10), $r_0(\bar{\boldsymbol{x}}) \geq r^*$. Therefore, the accumulation point $\bar{\boldsymbol{x}}$ is a globally optimal solution of the problem (10). □

3.2 Application to mean-variance portfolio optimization

We develop here a cutting-plane algorithm that is specialized for the performance of the M-V portfolio optimization. Similarly to (11), we rewrite the POP formulation (9) as follows:

$$\begin{array}{ll} \underset{(\boldsymbol{w},z)\in\mathbb{R}^I\times\mathbb{R}}{\text{minimize}} & (1-\lambda)z - \lambda \displaystyle\sum_{\boldsymbol{\alpha}:\,\sum_{i\in\mathcal{I}}\alpha_i = T} \mathcal{C}_2(\boldsymbol{\alpha})\boldsymbol{w}^{\boldsymbol{\alpha}} \\ \text{subject to} & \text{Var}(\boldsymbol{w}) - z \leq 0; \quad \displaystyle\sum_{i\in\mathcal{I}} w_i = 1; \;\; L_i \leq w_i \leq U_i, \quad i \in \mathcal{I}; \quad z \geq 0. \end{array} \tag{17}$$

Since $\text{Var}(\boldsymbol{w})$ (see (6)) is always nonnegative, we impose the nonnegative constraint on z.

Let us suppose that $(\bar{\boldsymbol{w}}, \bar{z})$ is an element of the following set:

$$\left\{(\boldsymbol{w}, z) \,\middle|\, \sum_{i\in\mathcal{I}} w_i = 1;\; L_i \leq w_i \leq U_i,\; i \in \mathcal{I};\;\; z \geq 0\right\}. \tag{18}$$

The gradient of (6) with respect to v_T^s is as follows:

$$G_s(\bar{\boldsymbol{w}}) := \frac{\partial}{\partial v_T^s}\left(\sum_{s'\in\mathcal{S}} P_{s'}(v_T^{s'})^2 - \left(\sum_{s'\in\mathcal{S}} P_{s'} v_T^{s'}\right)^2\right)\Bigg|_{v_T^s = v_T^s(\bar{\boldsymbol{w}}),\ s\in\mathcal{S}}$$

$$= 2P_s v_T^s(\bar{\boldsymbol{w}}) - 2P_s \sum_{s' \in \mathcal{S}} P_{s'} v_T^{s'}(\bar{\boldsymbol{w}}), \tag{19}$$

where

$$v_T^s(\bar{\boldsymbol{w}}) := V \prod_{t \in \mathcal{T}} \left(\sum_{i \in \mathcal{I}} R_{i,t}^s \bar{w}_i \right) \quad \text{for all } s \in \mathcal{S}. \tag{20}$$

Then, considering that

$$\begin{aligned}
&\mathrm{Var}(\bar{\boldsymbol{w}}) + \sum_{s \in \mathcal{S}} G_s(\bar{\boldsymbol{w}})(v_T^s - v_T^s(\bar{\boldsymbol{w}})) \\
&\overset{(4),(8)}{=} \mathrm{Var}(\bar{\boldsymbol{w}}) + \sum_{s \in \mathcal{S}} G_s(\bar{\boldsymbol{w}}) \left(\sum_{\boldsymbol{\alpha}: \sum_{i \in \mathcal{I}} \alpha_i = T} \mathcal{C}_1(\boldsymbol{\alpha}, s) \boldsymbol{w}^{\boldsymbol{\alpha}} - v_T^s(\bar{\boldsymbol{w}}) \right) \\
&\overset{(6),(19),(20)}{=} \sum_{\boldsymbol{\alpha}: \sum_{i \in \mathcal{I}} \alpha_i = T} \left(\sum_{s \in \mathcal{S}} G_s(\bar{\boldsymbol{w}}) \mathcal{C}_1(\boldsymbol{\alpha}, s) \right) \boldsymbol{w}^{\boldsymbol{\alpha}} - \mathrm{Var}(\bar{\boldsymbol{w}}),
\end{aligned} \tag{21}$$

we can show the following property:

Lemma 1 (*Underestimator of variance of portfolio value*)

$$\mathrm{Var}(\boldsymbol{w}) \geq \sum_{\boldsymbol{\alpha}: \sum_{i \in \mathcal{I}} \alpha_i = T} \left(\sum_{s \in \mathcal{S}} G_s(\bar{\boldsymbol{w}}) \mathcal{C}_1(\boldsymbol{\alpha}, s) \right) \boldsymbol{w}^{\boldsymbol{\alpha}} - \mathrm{Var}(\bar{\boldsymbol{w}}),$$

for all $\boldsymbol{w} \in \mathbb{R}^I$.

Proof To complete the proof, it is only necessary to show that (6) is a convex function in $(v_T^s)_{s \in \mathcal{S}}$ (see (13) and (21)). The following Hessian matrix of (6) with respect to $(v_T^s)_{s \in \mathcal{S}}$ is a positive semidefinite matrix:

$$\boldsymbol{H} := \begin{pmatrix} 2P_1(1-P_1) & -2P_1P_2 & \cdots & -2P_1P_S \\ -2P_1P_2 & 2P_2(1-P_2) & & \vdots \\ \vdots & & \ddots & -2P_{S-1}P_S \\ -2P_1P_S & \cdots & -2P_{S-1}P_S & 2P_S(1-P_S) \end{pmatrix},$$

because

$$\begin{aligned}
\frac{1}{2}\boldsymbol{x}^\top \boldsymbol{H} \boldsymbol{x} &= \sum_{s \in \mathcal{S}} P_s(1-P_s)x_s^2 - \sum_{s \in \mathcal{S}} \sum_{\substack{s' \in \mathcal{S} \\ s' \neq s}} P_s P_{s'} x_s x_{s'} \\
&\overset{(1)}{=} \sum_{s \in \mathcal{S}} P_s \left(\sum_{\substack{s' \in \mathcal{S} \\ s' \neq s}} P_{s'} \right) x_s^2 - \sum_{s \in \mathcal{S}} \sum_{\substack{s' \in \mathcal{S} \\ s' \neq s}} P_s P_{s'} x_s x_{s'}
\end{aligned}$$

$$
\begin{aligned}
&= \sum_{s\in\mathcal{S}} \sum_{\substack{s'\in\mathcal{S}\\ s'\neq s}} P_s P_{s'} (x_s^2 - x_s x_{s'}) \\
&= \sum_{s\in\mathcal{S}} \sum_{\substack{s'\in\mathcal{S}\\ s'> s}} P_s P_{s'} (x_s^2 - 2x_s x_{s'} + x_{s'}^2) \\
&= \sum_{s\in\mathcal{S}} \sum_{\substack{s'\in\mathcal{S}\\ s'> s}} P_s P_{s'} (x_s - x_{s'})^2 \overset{(1)}{\geq} 0 \quad \text{for all } \boldsymbol{x}\in\mathbb{R}^S.
\end{aligned}
$$

Therefore, (6) is a convex function in $(v_T^s)_{s\in\mathcal{S}}$. □

Thus, we derive the following inequality similarly to (14):

$$
z \geq \sum_{\boldsymbol{\alpha}:\, \sum_{i\in\mathcal{I}} \alpha_i = T} \left(\sum_{s\in\mathcal{S}} G_s(\bar{\boldsymbol{w}}) \mathcal{C}_1(\boldsymbol{\alpha}, s) \right) \boldsymbol{w}^{\boldsymbol{\alpha}} - \mathrm{Var}(\bar{\boldsymbol{w}}). \tag{22}
$$

If $(\bar{\boldsymbol{w}}, \bar{z})$ is an infeasible solution of the problem (17), that is, $\bar{z} < \mathrm{Var}(\bar{\boldsymbol{w}})$, then $(\boldsymbol{w}, z) := (\bar{\boldsymbol{w}}, \bar{z})$ does not satisfy the inequality (22). Moreover, the right-hand side of (22) is a global underestimator of $\mathrm{Var}(\boldsymbol{w})$ (see Lemma 1).

The algorithm for the M-V portfolio optimization problem (9) is described as follows:

Algorithm CPMV: Cutting-Plane Algorithm for the M-V Portfolio Optimization Problem (9)

Step 0. (Initialization) Let $\varepsilon \geq 0$ be a tolerance for optimality, K be the maximum number of iterations, and $\omega \geq \lceil T/2 \rceil$ be the relaxation order. Define $\mathcal{Z}$ as (18). Set the initial upper bound as $\mathrm{UB}_0 := \infty$. Set $k \leftarrow 1$.

Step 1. (Lower-Bound Estimation) Solve the following POP by using the SDP approach [15] with the relaxation order ω:

$$
\underset{(\boldsymbol{w},z)\in\mathbb{R}^I\times\mathbb{R}}{\text{minimize}}\ (1-\lambda)z - \lambda \sum_{\boldsymbol{\alpha}:\, \sum_{i\in\mathcal{I}} \alpha_i = T} \mathcal{C}_2(\boldsymbol{\alpha}) \boldsymbol{w}^{\boldsymbol{\alpha}} \quad \text{subject to } (\boldsymbol{w}, z) \in \mathcal{Z}. \tag{23}
$$

Let LB_k be the objective function value, and $(\bar{\boldsymbol{w}}^k, \bar{z}^k)$ be the solution of (23).

Step 2. (Upper-Bound Update) If $\mathrm{OF}(\bar{\boldsymbol{w}}^k) < \mathrm{UB}_{k-1}$, then $\mathrm{UB}_k := \mathrm{OF}(\bar{\boldsymbol{w}}^k)$ and $\hat{\boldsymbol{w}} \leftarrow \bar{\boldsymbol{w}}^k$. Otherwise, $\mathrm{UB}_k := \mathrm{UB}_{k-1}$.

Step 3. (Termination Conditions) If one of the following conditions is satisfied, then terminate the algorithm with the solution $\hat{\boldsymbol{w}}$:
$\langle \mathrm{a} \rangle$ $\mathrm{UB}_k - \mathrm{LB}_k \leq \varepsilon$, $\langle \mathrm{b} \rangle$ $\bar{z}^k \geq \mathrm{Var}(\bar{\boldsymbol{w}}^k)$, $\langle \mathrm{c} \rangle$ $k = K$.

Step 4. (Cut Generation) Set

$$
\mathcal{Z} \leftarrow \mathcal{Z} \cap \left\{ (\boldsymbol{w}, z) \,\middle|\, z \geq \sum_{\boldsymbol{\alpha}:\, \sum_{i\in\mathcal{I}} \alpha_i = T} \left(\sum_{s\in\mathcal{S}} G_s(\bar{\boldsymbol{w}}^k) \mathcal{C}_1(\boldsymbol{\alpha}, s) \right) \boldsymbol{w}^{\boldsymbol{\alpha}} - \mathrm{Var}(\bar{\boldsymbol{w}}^k) \right\},
$$

and $k \leftarrow k+1$. Return to Step 1.

It is clear from Lemma 1 that $\mathcal{Z}$ always contains the feasible region of the problem (17) in the algorithm. Hence, it follows that for all $k \geq 1$,

$$\mathrm{LB}_k \leq \text{ the global optimum of (17)} = \text{the global optimum of (9)} \leq \mathrm{UB}_k \leq \mathrm{OF}(\bar{\boldsymbol{w}}^k).$$

Moreover, if $(\bar{\boldsymbol{w}}^k, \bar{z}^k)$ satisfies the condition ⟨b⟩ in Step 3, then it is a feasible solution of the problem (17), and we have

$$\begin{aligned}\mathrm{OF}(\bar{\boldsymbol{w}}^k) &= (1-\lambda)\mathrm{Var}(\bar{\boldsymbol{w}}^k) - \lambda \sum_{\boldsymbol{\alpha}:\, \sum\limits_{i\in\mathcal{I}} \alpha_i = T} \mathcal{C}_2(\boldsymbol{\alpha})(\bar{\boldsymbol{w}}^k)^{\boldsymbol{\alpha}} \\ &\leq (1-\lambda)\bar{z}^k - \lambda \sum_{\boldsymbol{\alpha}:\, \sum\limits_{i\in\mathcal{I}} \alpha_i = T} \mathcal{C}_2(\boldsymbol{\alpha})(\bar{\boldsymbol{w}}^k)^{\boldsymbol{\alpha}} = \mathrm{LB}_k.\end{aligned}$$

Therefore, $\bar{\boldsymbol{w}}^k$ is an optimal solution of the problem (9). Although the condition ⟨b⟩ implies the condition ⟨a⟩ in theory, the condition ⟨b⟩ can occur without satisfying the condition ⟨a⟩ because of numerical instability. For the same reason it might happen that our algorithm has no improvement in the gap ⟨a⟩. However, this happens rarely and only when the gap is already small (for more details see Sect. 4). Therefore, if there is no improvement in the gap ⟨a⟩, we stop the algorithm and call this case ⟨d⟩.

Note that the maximal degree of monomials in the problem (23) is T while in the POP (9) is $2T$. Moreover, the CPMV algorithm has an advantage that the associated cut (22) does not contain a parameter λ. This means that even if we set λ to a different value, we can still use the cuts constructed for the previous value of λ. In computational experiments, the number of iterations has been reduced by taking advantage of this feature. Also, when $(\bar{\boldsymbol{w}}^k, \bar{z}^k)$ is not an optimal solution of (23) at Step 1, we set $\omega \leftarrow \omega + 1$ and restart the algorithm.

Other risk measures We can implement risk measures other than variance (see e.g., [14]) in the cutting-plane algorithm. For instance, minimizing Conditional Value-at-Risk (CVaR, [23]) is equivalent to minimizing the following function:

$$\begin{aligned}\mathrm{CVaR}(\boldsymbol{w}, a) &:= a + \frac{1}{1-\beta}\sum_{s\in\mathcal{S}} P_s \Psi\left(v_T^s - a\right) \\ &\overset{(4)}{=} a + \frac{1}{1-\beta}\sum_{s\in\mathcal{S}} P_s \Psi\left(V \prod_{t\in\mathcal{T}}\left(\sum_{i\in\mathcal{I}} R_{i,t}^s w_i\right) - a\right),\end{aligned}$$

where $\beta \in [0, 1)$ is a threshold parameter, and $\Psi : \mathbb{R} \to \mathbb{R}$ is a smoothing function of $\max\{0, \cdot\}$ (see e.g., [21]). The above function is convex in $(v_T^1, v_T^2, \ldots, v_T^S, a)$ (see [21, 23]). Moreover, all coherent risk measures [2] can be used in our cutting-plane algorithm because of their convexity.

4 Computational experiments

In this section, we use the following parameter values: the number of assets $I \in \{4, 7, 10\}$, the number of periods $T \in \{2, 4, 6\}$, and the number of scenarios $S \in$

{100, 1,000}. We set the initial wealth $V = 1$. The occurrence probability, P_s, is set to $1/|\mathcal{S}|$ for all $s \in \mathcal{S}$. The lower bound, L_i, and the upper bound, U_i, of the investment proportion are set to 0 and 0.5, respectively for each $i \in \mathcal{I}$. We choose for the trade-off parameter the following eight values $\lambda \in \{0.01, 0.1, 0.2, 0.3, 0.4, 0.5, 0.6, 0.99\}$. In the cutting-plane algorithm, we set the tolerance for optimality $\varepsilon = 10^{-5}$, and the maximum number of iterations $K = 30$. All computations were performed on a PC with a Core2 Duo CPU (1.40 GHz) and 2 GB memory. We used MATLAB 7.10.0 (R2010a) and a free MATLAB toolbox, GloptiPoly 3.6.1 [7], to formulate and solve POPs via the SDP relaxations (see [15]). In this toolbox, SeDuMi 1.3 [25] was used to solve SDP problems. We used the default parameters settings of GloptiPoly and SeDuMi. We also used a global optimization solver BARON [24], and a NLP solver CONOPT [4], via NEOS Server.[1] In BARON, a tolerance for optimality is set to the same value as in the cutting-plane algorithm, i.e., to 10^{-5}. In CONOPT, we do not set an initial solution, that is, CONOPT seeks the starting point itself.

Numerical data and notations in Tables 2, 3, 4 and 5 are as follows:

Rel.Order:	the relaxation order ω,
TotalCPU:	the total CPU time (in seconds),
CPUSDP:	CPU time for solving SDP relaxation problem (in seconds),
#Val.SDP:	the number of variables of the corresponding SDP in dual standard form,
SizeMat.:	the size of a semidefinite matrix in the corresponding SDP,
GapSDP:	the (average) duality gap associated with the obtained solution for SDP relaxation problem,
CPUCoef.	(in Table 2): CPU time for calculating $\mathcal{C}_2(\boldsymbol{\alpha})$ and $\mathcal{C}_5(\boldsymbol{\alpha})$ (in seconds),
CPUCoef.	(in Table 3): CPU time for calculating $\mathcal{C}_1(\boldsymbol{\alpha}, s)$ and $\mathcal{C}_2(\boldsymbol{\alpha})$ (in seconds),
#Iteration:	the number of iterations (i.e., k) in the cutting-plane algorithm,
#Ter.Con.:	the number of times each termination condition was satisfied, ($\langle$a$\rangle$, $\langle$b$\rangle$, $\langle$c$\rangle$, $\langle$d$\rangle$), see CPMV algorithm and explanations therein,
Ter.Con.:	the satisfied termination condition,
Opt.GAP:	the optimality gap, i.e., (the best upper bound) − (the best lower bound),
#Mem.Sho.:	the occurrence number of memory shortage,
Mem.Sho.:	"MS" is written in the case that there was a memory shortage, see Table 5,
OMS:	out of memory in SeDuMi, and
OMG:	out of memory in GloptiPoly.

For each pair of (I, T, S) we solve eight problems corresponding to different values of the trade-off parameter, λ. In Tables 2, 3 and 4, we show the average value of the eight problems in TotalCPU, CPUSDP, GapSDP and #Iteration, and the largest value of those in Opt.GAP.

Scenario generation. We have generated scenarios of total return, $R_{i,t}^s$, in a simple manner similar to [8]. We have first collected historical data of asset price from the

[1] http://www-neos.mcs.anl.gov

Table 1 Mean and variance of total returns

$\mathrm{MTR}(i) := \frac{1}{TS}\sum_{t\in\mathcal{T}}\sum_{s\in\mathcal{S}} R_{i,t}^s, \quad \mathrm{VTR}(i) := \frac{1}{TS}\sum_{t\in\mathcal{T}}\sum_{s\in\mathcal{S}} (R_{i,t}^s - \mathrm{MTR}(i))^2$

Asset #	1	2	3	4	5	6	7	8	9	10
MTR	1.0000	1.0056	1.0090	1.0110	1.0052	1.0067	1.0112	1.0032	1.0104	1.0121
VTR	0.0000	0.0009	0.0025	0.0035	0.0019	0.0019	0.0028	0.0005	0.0020	0.0032

Table 2 Numerical results of POP formulation (9)

$S = 100$			I			$S = 1{,}000$			I		
			4	7	10				4	7	10
T	2	Rel.Order	2	2	2	T	2	Rel.Order	2	2	2
		TotalCPU	0.6	3.0	26.3			TotalCPU	0.6	2.8	26.0
		CPUSDP	0.4	1.9	14.6			CPUSDP	0.4	1.7	14.3
		#Val.SDP	69	329	1,000			#Val.SDP	69	329	1,000
		SizeMat.	58	151	289			SizeMat.	58	151	289
		GapSDP	8.0E–09	2.8E–08	7.2E–08			GapSDP	6.7E–09	3.7E–08	6.3E–08
		CPUCoef.	0.0	0.1	0.3			CPUCoef.	0.0	0.1	0.3
	4	Rel.Order	4	4	4		4	Rel.Order	4	4	4
		TotalCPU	8.5	OMS	OMG			TotalCPU	8.6	OMS	OMG
		CPUSDP	7.4	–	–			CPUSDP	7.4	–	–
		#Val.SDP	494	6,434	–			#Val.SDP	494	6,434	–
		SizeMat.	353	2,013	–			SizeMat.	353	2,013	–
		GapSDP	1.2E–08	–	–			GapSDP	1.8E–08	–	–
		CPUCoef.	0.1	3.4	39.4			CPUCoef.	0.2	4.5	53.1
	6	Rel.Order	7	6	6		6	Rel.Order	7	6	6
		TotalCPU	1,223.2	OMG	OMG			TotalCPU	1,180.2	OMG	OMG
		CPUSDP	1,215.0	–	–			CPUSDP	1,172.1	–	–
		#Val.SDP	3,059	–	–			#Val.SDP	3,059	–	–
		SizeMat.	2,013	–	–			SizeMat.	2,013	–	–
		GapSDP	8.7E–08	–	–			GapSDP	1.0E–07	–	–
		CPUCoef.	1.0	93.0	3,384.4			CPUCoef.	1.4	130.6	4,639.6

Yahoo finance Japan.[2] Using these data, we estimated the mean vector $\boldsymbol{\mu} \in \mathbb{R}^{IT}$ and the variance-covariance matrix $\boldsymbol{\Sigma} \in \mathbb{R}^{IT\times IT}$ of total returns of asset $i \in \mathcal{I}$ at period $t \in \mathcal{T}$. Then, we generated scenarios of total return by drawing samples from a multivariate normal distribution with the estimated statistics $(\boldsymbol{\mu}, \boldsymbol{\Sigma})$. For reference, Table 1 shows characteristics of the total return.

[2]http://finance.yahoo.co.jp

Table 3 Numerical results of algorithm CPMV

$S = 100$			I			$S = 1{,}000$			I		
			4	7	10				4	7	10
T	2	Rel.Order	2	2	2	T	2	Rel.Order	2	2	2
		TotalCPU	13.0	65.6	960.9			TotalCPU	12.9	114.0	733.1
		#Iteration	3.4	5.1	7.9			#Iteration	3.3	6.8	6.4
		#Ter.Con.	(8, 0, 0, 0)	(8, 0, 0, 0)	(8, 0, 0, 0)			#Ter.Con.	(8, 0, 0, 0)	(8, 0, 0, 0)	(8, 0, 0, 0)
		Opt.Gap	8.8E–06	9.9E–06	9.9E–06			Opt.Gap	9.4E–06	1.0E–05	9.5E–06
		GapSDP	2.2E–07	4.9E–07	9.4E–07			GapSDP	8.4E–08	4.5E–07	4.7E–07
		CPUCoef.	0.0	0.0	0.0			CPUCoef.	0.0	0.0	0.0
	4	Rel.Order	2	2	2		4	Rel.Order	2	2	2
		TotalCPU	5.7	60.6	498.7			TotalCPU	11.3	68.8	616.7
		#Iteration	2.9	7.4	6.4			#Iteration	5.0	8.6	7.8
		#Ter.Con.	(6, 0, 0, 2)	(1, 0, 0, 7)	(5, 0, 0, 3)			#Ter.Con.	(7, 1, 0, 0)	(2, 4, 0, 2)	(6, 0, 0, 2)
		Opt.Gap	1.4E–04	9.0E–05	1.2E–04			Opt.Gap	1.7E–05	1.1E–04	3.2E–05
		GapSDP	3.4E–07	9.8E–07	1.4E–06			GapSDP	2.5E–07	1.0E–06	1.1E–06
		Rel.Order	3	3	3			Rel.Order	3	3	3
		TotalCPU	34.2	9,078.4	OMG			TotalCPU	49.4	5,610.0	OMG
		#Iteration	3.5	9.5	–			#Iteration	4.6	6.6	–
		#Ter.Con.	(8, 0, 0, 0)	(3, 1, 0, 4)	–			#Ter.Con.	(7, 1, 0, 0)	(6, 2, 0, 0)	–
		Opt.Gap	9.4E–06	5.7E–05	–			Opt.Gap	1.2E–05	7.3E–05	–
		GapSDP	8.7E–08	4.1E–07	–			GapSDP	1.6E–07	2.2E–07	–
		CPUCoef.	0.0	0.4	2.9			CPUCoef.	0.1	0.5	3.5
	6	Rel.Order	3	3	3		6	Rel.Order	3	3	3
		TotalCPU	29.9	6,743.2	OMG			TotalCPU	46.3	5,215.4	OMG
		#Iteration	4.0	10.4	–			#Iteration	6.0	7.6	–
		#Ter.Con.	(8, 0, 0, 0)	(1, 6, 0, 1)	–			#Ter.Con.	(8, 0, 0, 0)	(2, 4, 0, 2)	–
		Opt.Gap	9.4E–06	5.9E–05	–			Opt.Gap	9.6E–06	1.7E–04	–
		GapSDP	4.2E–07	1.4E–06	–			GapSDP	3.9E–07	1.0E–06	–
		Rel.Order	4	4	4			Rel.Order	4	4	4
		TotalCPU	329.2	OMG	OMG			TotalCPU	483.1	OMG	OMG
		#Iteration	3.9	–	–			#Iteration	5.4	–	–
		#Ter.Con.	(8, 0, 0, 0)	–	–			#Ter.Con.	(8, 0, 0, 0)	–	–
		Opt.Gap	8.7E–06	–	–			Opt.Gap	1.0E–05	–	–
		GapSDP	8.0E–08	–	–			GapSDP	1.7 E–07	–	–
		CPUCoef.	0.5	31.9	1,094.0			CPUCoef.	0.8	41.0	1,195.6

4.1 Numerical results of POP approaches

Numerical results of the POP formulation (9) and the cutting-plane algorithm are shown in Tables 2 and 3, respectively. We use the software GloptiPoly to solve the POP formulation (9) and the problem (23) in the cutting-plane algorithm. We have also tested other MATLAB toolboxes for solving POPs, i.e., SOSTOOLS[3] and SparsePOP [28]. In most cases SparsePOP solved the POP formulation a bit faster than GloptiPoly. SparsePOP failed to provide an optimal solution of the problem (23) with the smallest relaxation order $\omega = T/2$. GloptiPoly solved problems (9) and (23) faster than SOSTOOLS. For all these reasons, we omit results obtained by SOSTOOLS and SparsePOP.

Note that all solutions reported in Table 2 are globally optimal. It is clear that in the case of the POP formulation the number of scenarios has little impact on a CPU time. There, most of the total CPU time was consumed on solving SDP (see TotalCPU and CPUSDP in Table 2), while CPU time of calculating $\mathcal{C}_2(\boldsymbol{\alpha})$ and $\mathcal{C}_5(\boldsymbol{\alpha})$ was much shorter than TotalCPU and CPUSDP (see CPUCoef. in Table 2). However, only the problem involving four assets was solved when the number of periods was four or six, and it took about 1,200 seconds to solve the problem when $(I, T) = (4, 6)$.

To the contrary, by appropriately setting the relaxation order, the cutting-plane algorithm solved all the problems with satisfactory accuracy except when $(I, T) = (10, 6)$ (see Table 3). Our algorithm did not work well with the smallest relaxation order, $\omega = T/2$ when the number of periods was two. In other cases (i.e., when $T = 4$ or 6), a globally optimal solution of the problem (23) was always provided by the SDP relaxation regardless of whether $\omega = T/2$ or $\omega = T/2 + 1$. Note that the termination condition $\langle$c$\rangle$ was never satisfied, i.e., the number of iteration in the cutting-plane algorithm was always less than 30. Also, the algorithm has terminated several times due to the numerical instability. For instance, when $(I, T, S) = (7, 6, 1000)$ and $\omega = T/2$, the condition $\langle$b$\rangle$ was satisfied four times, and the algorithm has terminated two times because of the numerical issue $\langle$d$\rangle$. Although it is possible that the attained solution is not very good in such cases, the obtained optimality gap was sufficiently small (see worst-case optimality gap, Opt.GAP in Table 3).

Moreover, our numerical experiments show that by increasing the relaxation order from $\omega = T/2$ to $\omega = T/2 + 1$, we can reduce the duality gap for SDP relaxation, and the number of times the termination condition $\langle$d$\rangle$ is satisfied when $(I, T) = (4, 4)$ and $(7, 4)$ (see GapSDP and #Ter.Con. in Table 3). In addition, it is clear that GapSDP tends to be large in the problems that cause numerical instability. From these observations, we can see that numerical instability does not come from the algorithm framework but the quality of solutions of SDP relaxation problems, and that an effective remedy for numerical instability might be increasing the relaxation order. However, the larger relaxation order leads to the larger underlined SDP problem that needs to be solved. In the cutting-plane algorithm, CPU time for calculating $\mathcal{C}_1(\boldsymbol{\alpha}, s)$ and $\mathcal{C}_2(\boldsymbol{\alpha})$ was much shorter than the total CPU time (see CPUCoef. in Table 3).

4.2 Comparison with BARON and CONOPT

Numerical results of BARON and CONOPT are shown in Table 4, where four periods and 1,000 scenarios are considered. CPU times for solving problems by BARON

[3] http://www.cds.caltech.edu/sostools and http://www.mit.edu/~parrilo/sostools.

Table 4 Numerical results of BARON and CONOPT

BARON						CONOPT					
$S = 1{,}000$			I			$S = 1{,}000$			I		
			4	7	10				4	7	10
T	4	TotalCPU	804.2	5,031.7	7,537.8	T	4	TotalCPU	0.5	0.8	1.7
		Opt.Gap	3.4E–03	1.0E–05	1.2E–02			Opt.Gap	–	–	–
		#Mem.Sho.	4	0	5			#Mem.Sho.	0	0	0

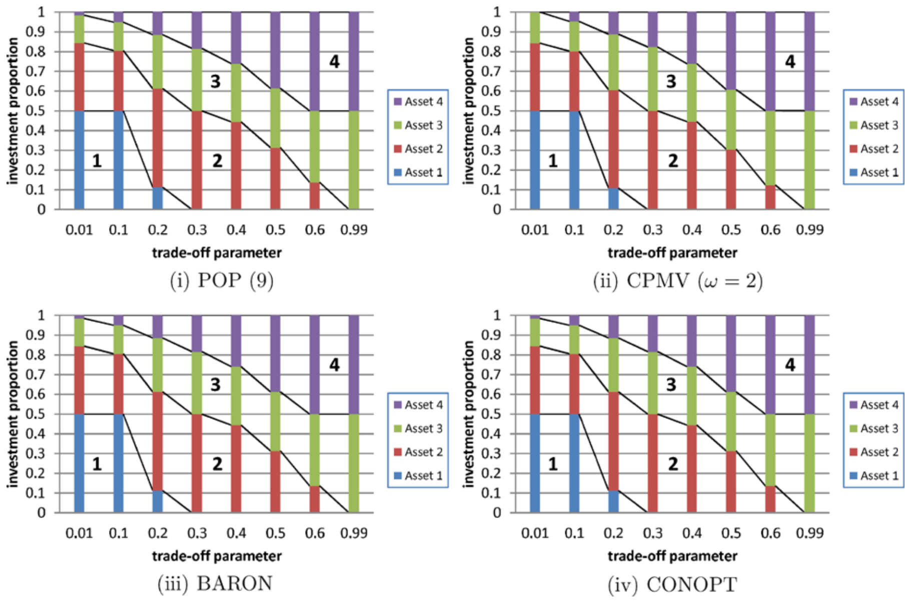
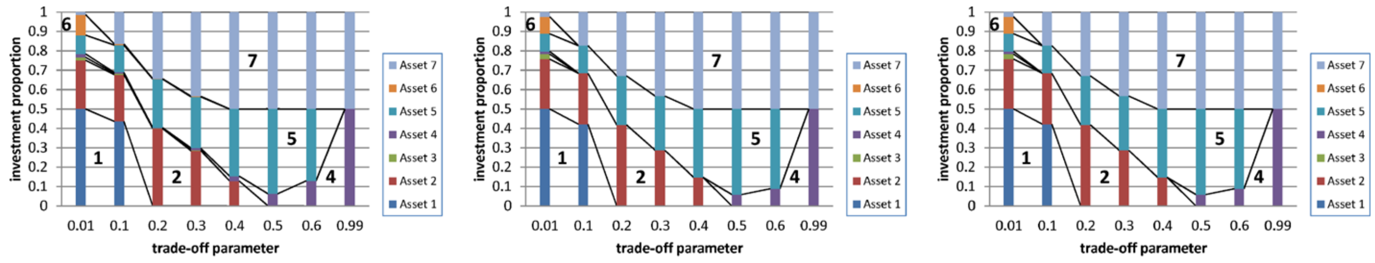


Fig. 2 Optimal investment proportion ($I = 4$, $T = 4$, $S = 1{,}000$)

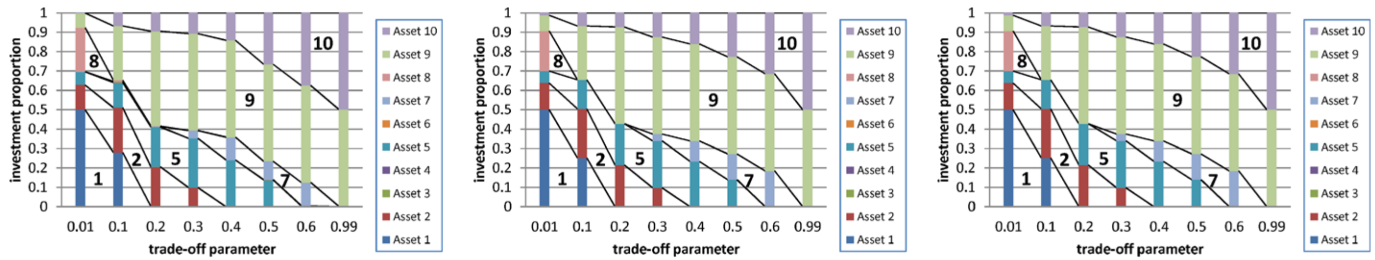
were very large in comparison to the cutting-plane algorithm with the smallest relaxation order. In some cases, BARON stopped due to the memory shortage and returned a locally optimal solution (see the last row in Table 4). However, CONOPT attained locally optimal solutions in very short time without leading to memory shortage.

In Figs. 2, 3 and 4, the optimal investment proportions provided by different approaches are shown. The difference between the investment proportions provided by BARON and those by CONOPT was always less than $1.1 \cdot 10^{-5}$ regardless of whether or not BARON caused memory shortage. POP formulation (9) was solved only when $I = 4$ ($T = 4$, $S = 1{,}000$), and in this case, the maximum difference in the obtained investment proportions between POP formulation and both BARON and CONOPT was $1.4 \cdot 10^{-3}$. The solutions of the cutting-plane algorithm were slightly different from others. For instance, the proportion in Asset 4 for $\lambda = 0.6$ was larger than other approaches (see Fig. 3). Also the proportion in Asset 7 for $\lambda = 0.5$ and 0.6 differs from other approaches (see Fig. 4).



(i) CPMV ($\omega = 2$) (ii) BARON (iii) CONOPT

Fig. 3 Optimal investment proportion ($I = 7$, $T = 4$, $S = 1{,}000$)



(i) CPMV ($\omega = 2$) (ii) BARON (iii) CONOPT

Fig. 4 Optimal investment proportion ($I = 10$, $T = 4$, $S = 1{,}000$)

Table 5 Cause for termination of CPMV ($\omega = 2$) and BARON ($T = 4$, $S = 1{,}000$)

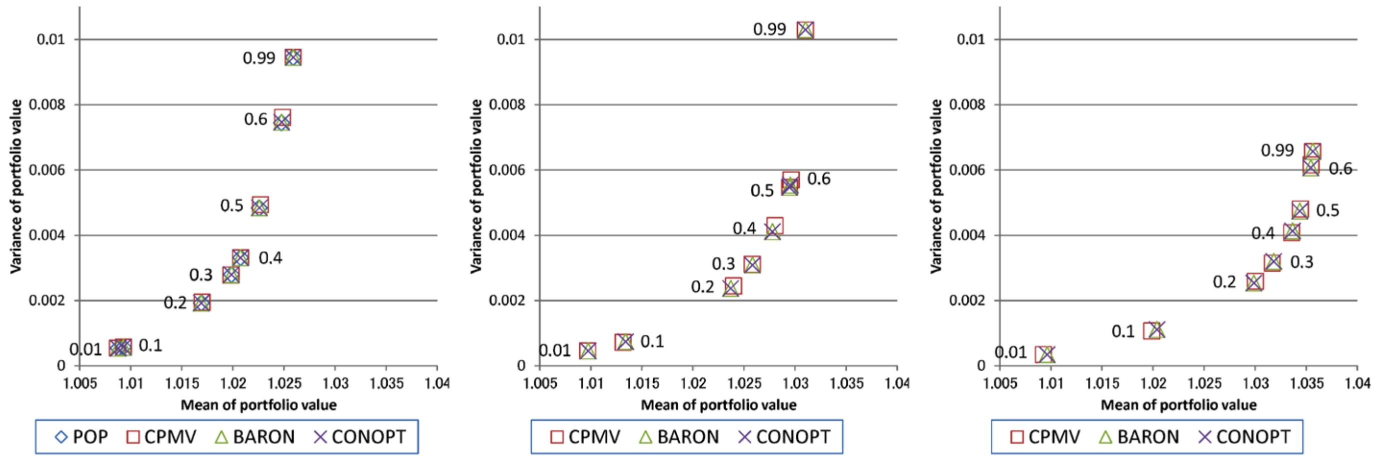
				Trade-off parameter							
				0.01	0.1	0.2	0.3	0.4	0.5	0.6	0.99
I	4	CPMV	Ter.Con.	⟨a⟩	⟨a⟩	⟨a⟩	⟨a⟩	⟨a⟩	⟨b⟩	⟨a⟩	⟨a⟩
			Opt.Gap.	9.3E–06	7.5E–06	7.0E–06	4.0E–06	9.0E–06	1.7E–05	8.3E–06	3.8E–07
		BARON	Mem.Sho.	MS	MS	MS	MS	–	–	–	–
			Opt.Gap.	3.4E–03	3.3E–03	2.9E–03	1.3E–03	1.0E–05	1.0E–05	1.0E–05	1.0E–05
	7	CPMV	Ter.Con.	⟨a⟩	⟨d⟩	⟨d⟩	⟨b⟩	⟨b⟩	⟨b⟩	⟨b⟩	⟨a⟩
			Opt.Gap.	6.9E–06	1.4E–05	2.2E–05	3.9E–05	1.5E–05	6.6E–05	1.1E–04	1.7E–06
		BARON	Mem.Sho.	–	–	–	–	–	–	–	–
			Opt.Gap.	1.0E–05	1.0E–05	1.0E–05	1.0E–05	1.0E–05	1.0E–05	1.0E–05	1.0E–05
	10	CPMV	Ter.Con.	⟨a⟩	⟨d⟩	⟨d⟩	⟨a⟩	⟨a⟩	⟨a⟩	⟨a⟩	⟨a⟩
			Opt.Gap.	8.8E–06	2.8E–05	3.2E–05	9.8E–06	6.7E–06	4.7E–06	9.5E–06	3.8E–06
		BARON	Mem.Sho.	MS	MS	MS	MS	MS	–	–	–
			Opt.Gap.	1.2E–02	1.0E–02	7.4E–03	3.9E–03	2.6E–04	1.0E–05	1.0E–05	1.0E–05

Cause for termination of the algorithm CPMV with $\omega = 2$ and BARON are shown in Table 5. It is found from Table 5 that BARON causes a memory shortage when the trade-off parameter $\lambda \leq 0.4$; and that the algorithm CPMV terminates because of the condition ⟨d⟩ only when $\lambda \leq 0.2$. Note that the small differences in the investment proportions mentioned above appear also in cases where the cutting-plane algorithm was not terminated due to the numerical instability. It seems to be difficult for our algorithm to always attain a high accuracy solution. As discussed above, this is due to solving one POP at each iteration of the algorithm. We expect that using the SDPA-GMP solver[4] under GloptiPoly will improve the quality of our solution. However, high precision computations provided by the mentioned solver result in high running times and therefore such solver is not acceptable for our iterative algorithm. As mentioned before, another way to improve a quality of the solution is to increase the relaxation order. However, the larger relaxation order leads to the larger underlined SDP problem that needs to be solved.

In Fig. 5, we show the efficient frontiers of the solutions provided by different approaches together with the value of the trade-off parameter. The horizontal axis and the vertical axis are mean and variance of the portfolio value, respectively. Some solutions of the cutting-plane algorithm were slightly different from solutions obtained by other solvers; i.e., the solutions that correspond to $(I, \lambda) = (4, 0.6), (7, 0.4), (7, 0.6)$ and $(10, 0.1)$. Those solutions satisfied termination conditions ⟨a⟩, ⟨b⟩, ⟨b⟩ and ⟨d⟩, respectively. Note that condition ⟨a⟩ is our tolerance for optimality, i.e., $\varepsilon = 10^{-5}$. However, it is also clear that solutions of the cutting-plane algorithm are not far from the frontiers of other approaches.

It was reported in [19] that an implemented NLP solver (for details see [19]) clearly failed to find a globally optimal solution of the problem (7). Meanwhile, our

[4] Available at http://sdpa.indsys.chuo-u.ac.jp/sdpa/software.html.



(i) $I = 4$ $(\omega = 2)$ (ii) $I = 7$ $(\omega = 2)$ (iii) $I = 10$ $(\omega = 2)$

Fig. 5 Efficient frontier ($T = 4$, $S = 1{,}000$)

numerical results indicate that CONOPT reached a globally optimal solution. This observation is in common with Fleten et al. [5]. In [5], the constant rebalanced portfolio optimization problem, in which the second order below-target risk is minimized, was solved using the NLP solver MINOS, and it was reported that MINOS always reached the same solution for an instance regardless of starting values.

5 Concluding remarks

We have developed the cutting-plane algorithm for solving the constant rebalanced portfolio optimization problem. Our algorithm, which is regarded as an extension of Kelley's convex cutting-plane algorithm, iteratively solves POPs by combining the SDP approach of [15] and valid cuts. The computational experiments show that the algorithm can solve large-size problems that can not be directly solved by global optimization solver over polynomials GloptiPoly [7]. This success is due to implementation of the reduced degree polynomials in the iterative algorithm. Our numerical results show that our algorithm provides solutions with adequate accuracy for practical purposes. Moreover, our algorithm is comparable to state-of-the-art global optimization solver BARON.

Furthermore, if there is an effective warm-starting approach for SDP, then our cutting-plane algorithm might be even more efficient by starting a SDP solver from the solution attained in the previous iteration.

A further direction of this study is to apply polynomial optimization approaches to other portfolio optimization problems. For instance, by taking into account skewness of the portfolio value as in [13], the problem can be formulated as a POP. Considering the current performance of SDP solvers it is difficult to solve POPs of high degree via SDP relaxations. However, SDP relaxation techniques and particularly large-scale SDPs are areas of active research, and thus, various POPs arising from portfolio optimization might be handled in the future.

Acknowledgements The first author is supported by the Grant-in-Aid for JSPS Fellows and the Excellent Young Researcher Overseas Visit Program of the JSPS.

References

1. Algoet, P.H., Cover, T.M.: Asymptotic optimality and asymptotic equipartition properties of log-optimum investment. Ann. Probab. **16**(2), 876–898 (1988)
2. Artzner, P., Delbaen, F., Eber, J.-M., Heath, D.: Coherent measures of risk. Math. Finance **9**(3), 203–228 (1999)
3. Cover, T.M.: An algorithm for maximizing expected log investment return. IEEE Trans. Inf. Theory **30**(2), 369–373 (1984)
4. Drud, A.S.: CONOPT—a large scale GRG code. ORSA J. Comput. **6**(2), 207–216 (1992)
5. Fleten, S.-E., Høyland, K., Wallace, S.W.: The performance of stochastic dynamic and fixed mix portfolio models. Eur. J. Oper. Res. **140**(1), 37–49 (2002)

6. Gatermann, K., Parrilo, P.A.: Symmetry groups, semidefinite programs, and sums of squares. J. Pure Appl. Algebra **192**(1–3), 95–128 (2004)
7. Henrion, D., Lasserre, J.B., Löfberg, J.: GloptiPoly 3: moments, optimization and semidefinite programming. Optim. Methods Softw. **24**(4–5), 761–779 (2009)
8. Hibiki, N.: Multi-period stochastic optimization models for dynamic asset allocation. J. Bank. Finance **30**(2), 365–390 (2006)
9. Jansson, L., Lasserre, J.B., Riener, C., Theobald, T.: Exploiting symmetries in SDP-relaxations for polynomial optimization. Preprint, Optimization Online (2006)
10. Kim, S., Kojima, M., Waki, H.: Generalized lagrangian duals and sums of squares relaxations of sparse polynomial optimization problems. SIAM J. Optim. **15**(3), 697–719 (2005)
11. Kojima, M., Kim, S., Waki, H.: A general framework for convex relaxation of polynomial optimization problems over cones. J. Oper. Res. Soc. Jpn. **46**(2), 125–144 (2003)
12. Kojima, M., Kim, S., Waki, H.: Sparsity in sums of squares of polynomials. Math. Program. **103**(1), 45–62 (2005)
13. Konno, H., Yamamoto, R.: A mean-variance-skewness model: algorithm and applications. Int. J. Theor. Appl. Finance **8**(4), 409–423 (2005)
14. Konno, H., Waki, H., Yuuki, A.: Portfolio optimization under lower partial risk measures. Asia-Pac. Financ. Mark. **9**(2), 127–140 (2002)
15. Lasserre, J.B.: Global optimization with polynomials and the problem of moments. SIAM J. Optim. **11**(3), 796–817 (2001)
16. Lasserre, J.B.: Convergent semidefinite relaxations in polynomial optimization with sparsity. SIAM J. Optim. **17**(3), 822–843 (2006)
17. Luenberger, D.G.: Investment Science. Oxford University Press, Oxford (1997)
18. Luenberger, D.G., Ye, Y.: Linear and Nonlinear Programming, 3rd edn. Springer, Berlin (2008)
19. Maranas, C.D., Androulakis, I.P., Floudas, C.A., Berger, A.J., Mulvey, J.M.: Solving long-term financial planning problems via global optimization. J. Econ. Dyn. Control **21**(8–9), 1405–1425 (1997)
20. Markowitz, H.: Portfolio selection. J. Finance **7**(1), 77–91 (1952)
21. Pang, J.-S., Leyffer, S.: On the global minimization of the value-at-risk. Optim. Methods Softw. **19**(5), 611–631 (2004)
22. Parrilo, P.A.: Semidefinite programming relaxations for semialgebraic problems. Math. Program. **96**(2), 293–320 (2003)
23. Rockafellar, R.T., Uryasev, S.: Conditional value-at-risk for general loss distributions. J. Bank. Finance **26**(7), 1443–1471 (2002)
24. Sahinidis, N.V., Tawarmalani, M.: A polyhedral branch-and-cut approach to global optimization. Math. Program. **103**(2), 225–249 (2005)
25. Sturm, J.F.: Using SeDuMi 1.02, a MATLAB toolbox for optimization over symmetric cones. Optim. Methods Softw. **11–12**, 625–653 (1999)
26. Takano, Y., Gotoh, J.: Constant rebalanced portfolio optimization under nonlinear transaction costs. Asia-Pac. Financ. Mark. **18**(2), 191–211 (2011)
27. Waki, H., Kim, S., Kojima, M., Muramatsu, M.: Sums of squares and semidefinite programming relaxations for polynomial optimization problems with structured sparsity. SIAM J. Optim. **17**(1), 218–242 (2006)
28. Waki, H., Kim, S., Kojima, M., Muramatsu, M., Sugimoto, H.: SparsePOP—a sparse semidefinite programming relaxation of polynomial optimization problems. ACM Trans. Math. Softw. **15**(2), 15 (2008)