Evolution on Trees: On the Design of an Evolution Strategy for Scenario-Based Multi-Period Portfolio Optimization under Transaction Costs

Hans-Georg Beyer[a], Steffen Finck[a,*], Thomas Breuer[a]

[a]*FH Vorarlberg University of Applied Sciences*

Abstract

Scenario-based optimization is a problem class often occurring in finance, planning and control. While the standard approach is usually based on (linear) stochastic programming, this paper develops an Evolution Strategy (ES) that can be used to treat also nonlinear planning problems arising from Value at Risk-constraints and not necessarily proportional transaction costs. The algorithm design is based on the covariance matrix self-adaptation ES (CMSA-ES). The evolution is performed on scenario trees where in each node specific constraints must be fulfilled. The design methodology is first applied to linear node constraints and compared with solutions obtained by linear programming (if feasibility of the linear system is ensured) and also with a commercial solution tool (in the case of non-feasibility of the linear system). In the general case with nonlinear node constraints we demonstrate both the potential of the ES designed and also its limitations. The latter are basically determined by the high dimensionality of the search spaces defined by the scenario trees.

Keywords: Evolution Strategy, Covariance Matrix Adaptation, Self-Adaptation, Portfolio Optimization

1. Introduction

This paper designs evolution strategies for discrete-time multi-period multi-asset portfolio optimisation problems with Value at Risk (VaR)-constraints and not necessarily proportional transaction costs. It is a follow-up of [1], which analysed the design of Evolution Strategies (ES) for one-period portfolio optimization problems under VaR-constraints without transaction costs, where it turned out that the algorithm design was challenging due to combination of seemingly simple constraints.

Here we deal with a stochastic control problem whose information and decision structure is defined on scenario trees. To our best knowledge, carrying out such optimizations using Evolutionary Algorithms (EAs) has not been considered so far. In the finance and operations research

*Corresponding address: FH Vorarlberg University of Applied Sciences, Hochschulstrasse 1, 6850 Dornbirn, Austria; Phone: +43 5572 7927122; Fax: +43 5572 7929510

Email addresses: `Hans-Georg.Beyer@fhv.at` (Hans-Georg Beyer), `Steffen.Finck@fhv.at` (Steffen Finck), `Thomas.Breuer@fhv.at` (Thomas Breuer)

literature, problems of this kind are treated by stochastic programming [2, 3]. However, due to the nonlinearities, the problem must be linearized to allow for the application of LP solvers. EAs allow for a much greater flexibility, which is required for the non-linearities arising from VaR-constraints and non-proportional transaction costs. The desired flexibility requires methods for efficiently handling the set of constraints accompanying such planning problems.

This paper is devoted to the design principles for ESs operating on tree structures encountered in such optimization problems. Sec. 2 introduces the problem in its general form. Sec. 3 deals with the problem in the special case of vanishing transaction costs and uses the standard CMSA-ES as an algorithmic skeleton to design an ES, the multi-period (mp) CMSA-ES, that operates on the tree structure of the optimization problem. Section 4 presents an experimental evaluation of the novel mpCMSA-ES. While the development of Sec. 3 is based on linear balance equations in the tree nodes, Sec. 5 extends the approach to the problem in its general form with non-linear constraints arising for example from non-proportional transaction costs. Section 6 summarizes the paper and provides an outlook.

2. The Optimization Problem

In many financial optimization models the uncertainty in asset prices (or other risk factors) is represented by a number of mass points forming a scenario tree. Such trees are often interpreted as approximating stochastic processes, but the interpretational issues involved in such approximations are of no concern to us. Consider an ordered directed tree the nodes of which are indexed by $n \in \{0, \dots, N\}$. The zero node is the root node. The predecessor node k of a node n is indexed by $k = \pi(n)$ (representing an adjacency list). There are N_L leaf nodes representing the terminal nodes having no child nodes. For sake of simplicity the leaf nodes are indexed by the consecutive numbers in $\mathbb{N}_L = \{N - N_L + 1, \dots, N\}$. For each leaf node there is a unique path leading to it. Denote by p_l, $l \in \mathbb{N}_L$, the probability of arriving a leaf node l.[1]

With various methods of scenario reduction the size of the scenario tree can be reduced [4, 5]. This is essential for numerical tractability, but in case the tree is too sparse arbitrage cannot be ruled out [6, 7]. A short selling constraint like (2) below still ensures the existence of a solution to the portfolio optimisation problem, but this solution will in general be biased.

Before transaction costs the wealth W of a node n is [2]

$$W_n := \boldsymbol{x}_n^{\mathrm{T}} \boldsymbol{\xi}_n = \sum_{m=1}^{M} (\boldsymbol{x}_n)_m (\boldsymbol{\xi}_n)_m, \tag{1}$$

where $\boldsymbol{x} = (x_1, \dots, x_M)^{\mathrm{T}}$ is the M-dimensional vector of the nodal portfolio weights x_m

$$\forall m = 1, \dots, M : \quad x_m \geq 0, \tag{2}$$

which amounts to a short selling constraint. The vector

$$\boldsymbol{\xi} = (\xi_1, \dots, \xi_M)^{\mathrm{T}}, \quad \forall m = 1, \dots, M : \ \xi_m > 0 \tag{3}$$

[1]Equivalently one could specify conditional probabilities for each edge of the tree. The conditional probabilities of each edge leaving the same node add up to one. In this formulation, p_l equals the product of conditional probabilities of all edges along the path leading from the root node to the leaf node l.

[2]Note, we use the notation $(\cdot)_m$ to refer to the mth component of a vector, i.e., $(\boldsymbol{x})_m \equiv x_m$.

contains the M asset prices in the respective node. Starting with an initial capital W_0 in the root node and the asset prices $\boldsymbol{\xi}_0$

$$W_0 = \boldsymbol{x}_0^{\mathrm{T}} \boldsymbol{\xi}_0, \tag{4}$$

the decision maker has to choose the portfolio weights $\boldsymbol{x}_0$ at time zero (being in the root node). At the next time step, a number of child scenarios with changed asset prices $\boldsymbol{\xi}_k$ are to consider. As a result of asset price changes in the transition from node $\pi(k)$ to note k, the value of the portfolio will change to $W_k = \boldsymbol{x}_{\pi(k)}^{\mathrm{T}} \boldsymbol{\xi}_k$ in the child node k. Now the decision maker can change the nodal portfolio weights in the child node constrained by the available wealth W_k

$$\forall k = 1, \ldots, N - N_L : \quad \boldsymbol{x}_k^{\mathrm{T}} \boldsymbol{\xi}_k = \boldsymbol{x}_{\pi(k)}^{\mathrm{T}} \boldsymbol{\xi}_k. \tag{5}$$

Additionally, we take into account transaction costs [8–13]. Let us call this cost function "B[uy]S[ell]". A simple option is

$$\mathrm{BS}_1(\boldsymbol{x}_\pi, \boldsymbol{x}, \boldsymbol{\xi}) := c \sum_{m=1}^{M} |(\boldsymbol{x}_\pi)_m - (\boldsymbol{x})_m| \, (\boldsymbol{\xi})_m \tag{6}$$

which accounts in a symmetric and proportional way for buy and sell transaction costs, assuming vanishing fixed costs of transactions. A somewhat more elaborated model might consider fixed costs $c_f \geq 0$ and different costs for buying $c_b \geq 0$ and selling $c_s \geq 0$

$$\begin{aligned} \mathrm{BS}_2(\boldsymbol{x}_\pi, \boldsymbol{x}, \boldsymbol{\xi}) :=& c_f \sum_{m=1}^{M} \overline{\delta}((\boldsymbol{x}_\pi)_m - (\boldsymbol{x})_m) \\ &+ c_b \sum_{m=1}^{M} \Theta((\boldsymbol{x})_m - (\boldsymbol{x}_\pi)_m) |(\boldsymbol{x}_\pi)_m - (\boldsymbol{x})_m| \, (\boldsymbol{\xi})_m \\ &+ c_s \sum_{m=1}^{M} \Theta((\boldsymbol{x}_\pi)_m - (\boldsymbol{x})_m) |(\boldsymbol{x}_\pi)_m - (\boldsymbol{x})_m| \, (\boldsymbol{\xi})_m, \end{aligned} \tag{7}$$

where $\overline{\delta}$ and Θ are defined in (A.9) and (A.10) in the Appendix.

Now, the decision maker may change the weights in the child nodes such that the wealth after a transaction equals the wealth before transaction minus the transaction costs. That is, in each inner node (i.e. those nodes not being root node or terminal nodes) a (linear) equality constraint must hold

$$\forall k = 1, \ldots, N - N_L : \quad \boldsymbol{x}_k^{\mathrm{T}} \boldsymbol{\xi}_k + \mathrm{BS}(\boldsymbol{x}_{\pi(k)}, \boldsymbol{x}_k, \boldsymbol{\xi}_k) = \boldsymbol{x}_{\pi(k)}^{\mathrm{T}} \boldsymbol{\xi}_k \tag{8}$$

The last portfolio choices are made at the predecessors of the leaf nodes. At the leaf nodes themselves a final wealth value is realised.

Thus a trading strategy $\mathbf{X}$ is described by a matrix made of the $N - N_L + 1$ non-terminal node decision vectors $\boldsymbol{x}_k$ (being column vectors)

$$\mathbf{X} = \big(\boldsymbol{x}_0, \ldots, \boldsymbol{x}_{N-N_L}\big), \tag{9}$$

which satisfy the budget constraints (8). The expected final wealth is $\sum_{l=N-N_L+1}^{N} p_l W_l$, where $W_l = \boldsymbol{x}_{\pi(l)}^{\mathrm{T}} \boldsymbol{\xi}_l$. The objective is to maximise this expected wealth at the end of the planning horizon,

$$f(\mathbf{X}) := \sum_{l=N-N_L+1}^{N} p_l \, \boldsymbol{x}_{\pi(l)}^{\mathrm{T}} \boldsymbol{\xi}_l \longrightarrow \max_{\boldsymbol{x}_0, \ldots, \boldsymbol{x}_{N-N_L}} . \tag{10}$$

In addition to the budget constraints (8) let us impose a risk constraint on the admissible trading strategies. Regulators [14] require banks to hold at the end of the trading day sufficient economic capital defined in terms of Value at Risk (VaR).[3] These regulatory VaR-constraints translate into the requirement that at least an α-fraction of final wealth values W_l are greater than (or equal to) a κ-multiple of the initial capital W_0. The function $\eta(\mathbf{X})$ sums all p_l for which this holds

$$\eta(\mathbf{X}) := \left\{ \sum_l p_l \mid l \in \mathbb{N}_L \ \wedge \ W_l \geq \kappa W_0 \right\}, \tag{11}$$

resulting in the non-linear inequality constraint

$$\eta(\mathbf{X}) \ \geq \ \alpha. \tag{12}$$

Let us summarize the optimization problem

$$\sum_{l=N-N_L+1}^{N} p_l \boldsymbol{x}_{\pi(l)}^{\mathrm{T}} \boldsymbol{\xi}_l \longrightarrow \max_{\mathbf{X}}, \tag{13a}$$

s. t.

$$\forall m, n: \ (\boldsymbol{x}_n)_m \geq 0, \tag{13b}$$

$$\forall k = 1, \ldots, N - N_L :$$

$$\boldsymbol{x}_k^{\mathrm{T}} \boldsymbol{\xi}_k + \mathrm{BS}(\boldsymbol{x}_{\pi(k)}, \boldsymbol{x}_k, \boldsymbol{\xi}_k) = \boldsymbol{x}_{\pi(k)}^{\mathrm{T}} \boldsymbol{\xi}_k \tag{13c}$$

$$W_0 = \boldsymbol{x}_0^{\mathrm{T}} \boldsymbol{\xi}_0, \tag{13d}$$

$$\eta(\mathbf{X}; \kappa, W_0) \ \geq \ \alpha \tag{13e}$$

where $\mathbf{X}$ is the collection of (nodal) $\boldsymbol{x}_k$-vectors ($k = 0, \ldots, N - N_L$) to be optimized.

3. CMSA-ES Design

In this and the next section we specialize to the case where the transaction costs vanish ($c_f = c_b = c_s = 0$), i.e., replacing (13c) with (5). This restriction will be lifted in Section 5.

3.1. The mpCMSA-ES Algorithm

The design of the ES is governed by the experiences gained in the development of the constraint CMSA-ES for portfolio optimization in [1]:

1. The linear constraints (5) will be fulfilled by the mutation process.
2. The non-linear constraint (13e) will be handled by the selection process which is based on a two-component vector fitness function as have been used in [1].

The first item concerning the mutation process will be specifically designed for evolution on trees as they appear in multi-periodic stochastic programs. The details will be presented in the following subsections. At this point we will discuss the CMSA-ES algorithm for multi-periodic problems, the mpCMSA-ES, as a whole in Fig. 1.

[3]Determining capital requirements defined in terms of some coherent or at least convex risk measure would be more satisfactory, see [15] and [16]. Actually, a constraint in the coherent risk measure CVaR would be easier to handle than a VaR-constraint, since it leads to a linear problem [17]. Furthermore, it would be more satisfactory to have a full-fledged capital requirement for processes rather than for the final wealth at the end of the trading period, since new information and transactions arise during the trading period [3, 18–20]. But we take the current regulatory framework as given.

```
$(\mu/\mu_I, \lambda)$-mpCMSA-ES

Initialize$\big(\mu, \lambda, \mathbf{X}^{(0)}, \sigma, \hat{\sigma}, \tau, \tau_c, G, g_{\text{stop}}, \sigma_{\text{stop}}, \epsilon, \mathbf{C}\big)$   (L1)
$\mathfrak{a}_{\text{bsf}} \leftarrow \big(f(\mathbf{X}^{(0)}), \nu(\mathbf{X}^{(0)}), \mathbf{X}^{(0)}, \sigma, \mathbf{0}\big)$   (L2)
$g \leftarrow 0$   (L3)
Repeat
  For $l \leftarrow 1$ To $\lambda$
    $\tilde{\sigma}_l \leftarrow \sigma \mathrm{e}^{\tau \mathcal{N}_l(0,1)}$   (L4)
    $\boldsymbol{z}_l \leftarrow \det(\mathbf{C})^{-\frac{1}{2K}} \sqrt{\mathbf{C}}\, \boldsymbol{\mathcal{N}}_l(\mathbf{0}, \mathbf{I})$   (L5)
    $\mathbf{Z}_l \leftarrow$ VectorToMatrix$(\boldsymbol{z}_l, M-1)$   (L6)
    $\tilde{\mathbf{X}}_l \leftarrow$ Mutate$(\mathbf{X}^{(g)}, \mathbf{Z}_l, \tilde{\sigma}_l;\ \mathbf{\Xi}, W_0)$   (L7)
    $\tilde{\nu}_l \leftarrow \nu(\tilde{\mathbf{X}}_l)$   (L8)
    $\tilde{f}_l \leftarrow f(\tilde{\mathbf{X}}_l)$   (L9)
    $\tilde{\mathbf{Z}}_l \leftarrow \frac{1}{\tilde{\sigma}_l}$ DropLastRow$\big(\tilde{\mathbf{X}}_l - \mathbf{X}^{(g)}\big)$   (L10)
    $\tilde{\boldsymbol{z}}_l \leftarrow$ MatrixToVector$(\tilde{\mathbf{Z}}_l)$   (L11)
    $\tilde{\mathfrak{a}}_l \leftarrow \big(\tilde{f}_l, \tilde{\nu}_l, \tilde{\mathbf{X}}_l, \tilde{\sigma}_l, \tilde{\boldsymbol{z}}_l\big)$   (L12)
    $\mathfrak{a}_{\text{bsf}} \leftarrow \begin{cases} \tilde{\mathfrak{a}}_l, & \text{if } \tilde{\mathfrak{a}}_l \succ \mathfrak{a}_{\text{bsf}} \\ \mathfrak{a}_{\text{bsf}}, & \text{otherwise} \end{cases}$   (L13)
  End
  RankOffspringPopulation$(\tilde{\mathfrak{a}}_1, \ldots, \tilde{\mathfrak{a}}_\lambda)$   (L14)
  $g \leftarrow g + 1$   (L15)
  $\mathbf{X}^{(g)} \leftarrow \langle \tilde{\mathbf{X}} \rangle$   (L16)
  $\sigma \leftarrow \begin{cases} \langle \tilde{\sigma} \rangle, & \text{if } \langle \tilde{\sigma} \rangle \le \hat{\sigma} \\ \hat{\sigma}, & \text{otherwise} \end{cases}$   (L17)
  $\mathbf{C} \leftarrow \left(1 - \frac{1}{\tau_c}\right)\mathbf{C} + \frac{1}{\tau_c}\langle \tilde{\boldsymbol{z}} \tilde{\boldsymbol{z}}^{\mathrm{T}} \rangle$   (L18)
Until$\big(\ g \ge g_{\text{stop}} \ \vee\ \sigma < \sigma_{\text{stop}} \ \vee\ \|\mathbf{X}^{(g)} - \mathbf{X}^{(g-G)}\| < \epsilon\ \big)$   (L19)
```

Figure 1: Pseudocode of the CMSA-ES for multi-periodic problems (mp) with elite conservation in the "best-so-far" individual $\mathfrak{a}_{\text{bsf}}$.

The basic idea of covariance matrix self-adaptation ES (CMSA-ES) has been introduced in [21] as an alternative to the standard CMA-ES [22]. In contrast to CMA-ES, the CMSA-ES uses mutative self-adaptation (SA) to evolve the general mutation strength σ. While showing somewhat lower performance on standard test beds compared to CMA-ES, the CMSA-ES has several advantages that makes it interesting for algorithm engineering:

- The cumulative step-size adaptation (CSA) as used in CMA-ES fails on problems in constrained search spaces if the optimum is on the boundary of the feasible region [23]. There are currently no design rules to adapt CSA to such situations.
- The mutative self-adaptation is a general and simple principle that does not rely on pecu-

liarities of the search space.

- The CMSA-ES does only need two time constants to be chosen.

The CMSA-ES in Fig. 1 uses μ parental individuals and generates λ offspring individuals, $\lambda > \mu$. As in each evolutionary algorithm, the parameters $\mathbf{X}$, Eq. (9), to be optimized must be initialized. The initialization details of the $M \times (N + 1 - N_L)$ matrix $\mathbf{X}$ will be discussed in conjunction with the mutation operation in the next subsection. The initialization of the mutation strength σ and the maximally admissible mutation strength $\hat{\sigma}$ are discussed below. The covariance matrix describing the distribution of the mutations is chosen initially as a $K \times K$ identity matrix where

$$K = (M - 1) \cdot (N + 1 - N_L). \tag{14}$$

Further details of the initialization will be discussed in consecutive sections.

Darwinian evolution acts on the individual level. An individual $\mathfrak{a}$ in mpCMSA-ES comprises the goal function value $f(\mathbf{X})$, Eq. (10); the so-called violation probability $\nu(\mathbf{X})$, Eq. (36) below; the object parameters $\mathbf{X}$, Eq. (9); the mutation strength σ; and the K-dimensional mutation direction vector $\boldsymbol{z}$. The ES developed uses elite conservation. That is, one keeps track of the best-so-far individual $\mathfrak{a}_{\text{bsf}}$ which is initialized in Line 2 in Fig. 1. The ES is run over at most g_{stop} generations where g is the generation counter. Further stopping criteria are given in Line 19 being the minimal mutation strength σ_{stop} and the minimal object parameter change ϵ within G generations.

In the evolutionary loop, λ offspring are generated through the Lines 4–12. In Line 13 an update of the best-so-far individual takes place if the currently generated offspring is better than the current best-so-far individual. The order relation "$\succ$" (indicating the "better" one to the left) used in Line 13 and also in 14 will be discussed in detail in Section 3.3.

The generation of a single offspring starts with the mutation of the parental mutation strength σ by a log-normally distributed random number. For the learning parameter τ the standard choice [21]

$$\tau = \frac{1}{\sqrt{2K}} \tag{15}$$

is used.

In Line 5 the *unrestricted* mutation direction $\boldsymbol{z}$ is generated using an ellipsoid-volume conserving transformation of a sample from a $(\mathbf{0}, \mathbf{I})$ normally distributed, K-dimensional random vector (i.e., all K components of $\boldsymbol{\mathcal{N}}$ are $\mathcal{N}(0, 1)$ iid). In Line 6 the $\boldsymbol{z}$-vector is partitioned into a $\mathbf{Z}$ matrix consisting of column vectors of dimension $M - 1$ (there are $N + 1 - N_L$ such vectors). Each of these column vectors is transformed into an $\boldsymbol{x}$ vector of dimension M by the mutation operation in Line 7. The mutation operation is applied to the parental $\mathbf{X}$. The details of this transformation that has to respect the constraints (13b) and (5) will be given in Section 3.2. As a result one obtains a set of $\boldsymbol{x}$ vectors collected in the $\tilde{\mathbf{X}}$ matrix being the offspring's object parameters. The fitness (goal function f and violation count ν) of the offspring is calculated in Lines 8 and 9.

Due to the inequality constraints (13b), there are cases where the mutational change $\tilde{\mathbf{X}} - \mathbf{X}$ is not in direction of the originally generated $\mathbf{Z}$. This must be taken into account when updating the covariance matrix $\mathbf{C}$. To this end, a "corrected" mutation direction $\tilde{\mathbf{Z}}_l$ is calculated from the difference $\tilde{\mathbf{X}}_l - \mathbf{X}$ by taking the first $M - 1$ components of the difference vectors into account in Line 10. The division by $\tilde{\sigma}_l$ ensures the scale-invariance. Stacking the columns of the resulting matrix on top of each other yields the covariance matrix update vector $\tilde{\boldsymbol{z}}$ in Line 11. Line 12 completes the offspring.

Having generated λ offspring, the population is ranked according to a lexicographic order (Line 14, for details, see Section 3.3) and the μ best offspring individuals are used to generate the new parental state in Line 16 and 17. The calculation of the new parental state is done in accordance with the standard CMSA-ES [21]: The object parameters are obtained by mean value calculation for each single x component of the selected (i.e. parental) μ individuals. The same is performed for the individual mutation strengths in Line 17. However, the resulting mutation strength is checked against a maximal value $\hat{\sigma}$ to avoid uncontrolled σ divergence. A reasonable choice for $\hat{\sigma}$ is given by

$$\hat{\sigma} = \frac{W_0}{\|\boldsymbol{\xi}_0\|}. \tag{16}$$

This value can also be used as initial σ value.

The admissibility of the recombination operation in Line 16 will be discussed briefly. First note, that $\tilde{\mathbf{X}}$ is made of the $N - N_L + 1$ M-dimensional column vectors $\tilde{\boldsymbol{x}}_k$. Recombination is done for each node k individually by centroid calculation over the best μ of the λ offspring individuals, denote by $\tilde{\boldsymbol{x}}_k^{l;\lambda}$:

$$\langle \boldsymbol{x}_k \rangle = \frac{1}{\mu} \sum_{l=1}^{\mu} \tilde{\boldsymbol{x}}_k^{l;\lambda}. \tag{17}$$

The resulting centroid $\langle \boldsymbol{x}_k \rangle$ must fulfill (13b) for each of its M components. This is guaranteed because each of the offspring individuals is generated such that its components $(\tilde{\boldsymbol{x}}_k^{l;\lambda})_m \geq 0$ (for the algorithmic realization, see below). Therefore, the sum of the components must be ≥ 0 as well, i.e., each of the components in (17) is ≥ 0. Additionally, the equality constraint (5) must be ensured. First note that $\boldsymbol{\xi}_k^{\mathrm{T}} \tilde{\boldsymbol{x}}_k^{l;\lambda} = W_k$ does hold for each offspring due to the design of the mutation operator (for details see Section 3.2). Take the sum over the equality constraints of the μ best offspring individuals

$$\frac{1}{\mu} \sum_{l=1}^{\mu} \boldsymbol{\xi}_k^{\mathrm{T}} \tilde{\boldsymbol{x}}_k^{l;\lambda} = \frac{1}{\mu} \sum_{l=1}^{\mu} W_k = W_k. \tag{18}$$

Exchanging order of summation and scalar product in the lhs of (18) yields

$$\boldsymbol{\xi}_k^{\mathrm{T}} \frac{1}{\mu} \sum_{l=1}^{\mu} \tilde{\boldsymbol{x}}_k^{l;\lambda} = \boldsymbol{\xi}_k^{\mathrm{T}} \langle \boldsymbol{x}_k \rangle = W_k. \tag{19}$$

As one can see, (8) is also fulfilled for the recombinant $\langle \boldsymbol{x}_k \rangle$.

Finally, Line 18 performs the covariance matrix update as developed in [21]. The time constant is given by

$$\tau_{\mathrm{C}} = 1 + \frac{(K+1)K}{2\mu}. \tag{20}$$

3.2. *Initialization of* $\mathbf{X}$ *and Mutation Operation*

Initialization and mutation have to respect the equation constraints (5) and the positivity condition (13b). Random initialization ensuring $x_m \geq 0$ can be easily obtained by sampling from the uniform $(0,1]$ distribution. However, this does not ensure the validity of (5) and (13d). The latter can be ensured by rescaling an M-dimensional random vector $\boldsymbol{u}$

$$\boldsymbol{u} := (\mathrm{u}(0,1],\ldots,\mathrm{u}(0,1])^{\mathrm{T}}, \tag{21}$$

where $0 < \mathrm{u}(0,1] \leq 1$ holds for each uniformly distributed random component in (21). Introducing the rescaling factor β_0 such that $\boldsymbol{x}_0 = \beta_0 \boldsymbol{u}_0$, Eq. (13d) becomes $\beta_0 \boldsymbol{u}_0^{\mathrm{T}} \boldsymbol{\xi}_0 \overset{!}{=} W_0$. Solving for β_0 yields $\beta_0 = W_0 / \boldsymbol{u}_0^{\mathrm{T}} \boldsymbol{\xi}_0$ and therefore

$$\boldsymbol{x}_0 = \frac{W_0}{\boldsymbol{u}_0^{\mathrm{T}} \boldsymbol{\xi}_0} \boldsymbol{u}_0. \tag{22}$$

Now one can perform a pre-order walk through the tree to calculate $\boldsymbol{x}_k$ vectors fulfilling (5) at node k. Since the predecessor node vector $\boldsymbol{x}_{\pi(k)}$ is already known, we have to demand $\beta_k \boldsymbol{u}_k^{\mathrm{T}} \boldsymbol{\xi}_k \overset{!}{=} \boldsymbol{x}_{\pi(k)}^{\mathrm{T}} \boldsymbol{\xi}_k$. Resolving for β_k and reinserting in $\boldsymbol{x}_k = \beta_k \boldsymbol{u}_k$ yields

$$\boldsymbol{x}_k = \frac{\boldsymbol{x}_{\pi(k)}^{\mathrm{T}} \boldsymbol{\xi}_k}{\boldsymbol{u}_k^{\mathrm{T}} \boldsymbol{\xi}_k} \boldsymbol{u}_k. \tag{23}$$

This calculation is to be done for all inner nodes to complete the initialization of $\mathbf{X}$.

The idea of performing the initialization by pre-order walk through the tree can also be transferred to the mutation process. That is, the mutation is performed stepwise starting at the root node. For sake of simplicity, we consider the generation of a single offspring dropping the individual index l in Fig. 1. Each column in the $\mathbf{Z}$ matrix in Line 6 contains the $(M-1)$-dimensional mutation direction for the respective node (e.g., the first column $\boldsymbol{z}_0$ may correspond to the root node, etc.). The reason why these column vectors $\boldsymbol{z}_k$ are only $(M-1)$-dimensional is due to the fact that the constraints (13b) and (5) reduce the effective degree of freedom in each of these nodes by one. That is, $(M-1)$ z-components are already sufficient to fulfill the respective constraint (see below).

First, consider the root node. The mutation direction is $\boldsymbol{z}_0$ and the parental state is $\boldsymbol{x}_0$. Mutation is generally done by adding a length-scaled mutation vector $\tilde{\sigma}\boldsymbol{s}$ to the parental state. The factor $\tilde{\sigma}$ is also referred to as the mutation strength. Demanding (13d) one gets

$$(\boldsymbol{x}_0 + \tilde{\sigma}\boldsymbol{s}_0)^{\mathrm{T}} \boldsymbol{\xi}_0 \overset{!}{=} W_0 \quad \Rightarrow \quad \boldsymbol{s}_0^{\mathrm{T}} \boldsymbol{\xi}_0 \overset{!}{=} 0, \tag{24}$$

since $\boldsymbol{x}_0^{\mathrm{T}} \boldsymbol{\xi}_0 = W_0$ due to initialization. Writing the rhs of (24) component-wise

$$0 \overset{!}{=} \sum_{m=1}^{M-1} (\boldsymbol{s}_0)_m (\boldsymbol{\xi}_0)_m + (\boldsymbol{s}_0)_M (\boldsymbol{\xi}_0)_M, \tag{25}$$

one sees that this equation can already be fulfilled by fixing the first $(M-1)$ components of $\boldsymbol{s}$. The Mth component follows by resolving for $(\boldsymbol{s}_0)_M$

$$(\boldsymbol{s}_0)_M = -\frac{1}{(\boldsymbol{\xi}_0)_M} \sum_{m=1}^{M-1} (\boldsymbol{s}_0)_m (\boldsymbol{\xi}_0)_m. \tag{26}$$

Identifying the first $(M-1)$ s-components with the components of the $\boldsymbol{z}_0$-vector, one obtains a mutation vector that fulfills (13d)

$$\boldsymbol{s}_0 = \left((\boldsymbol{z}_0)_1, \ldots, (\boldsymbol{z}_0)_{M-1}, -\frac{1}{(\boldsymbol{\xi}_0)_M} \sum_{m=1}^{M-1} (\boldsymbol{z}_0)_m (\boldsymbol{\xi}_0)_m \right)^{\mathrm{T}}. \tag{27}$$

Now the mutation in the root node can be performed resulting in a vector $\boldsymbol{v}_0$

$$\boldsymbol{v}_0 = \boldsymbol{x}_0 + \tilde{\sigma}\boldsymbol{s}_0. \tag{28}$$

This vector, however, can violate the positivity condition (13b). In such cases a repair must be made. Similar to the approach taken in [1], this repair should only minimally change the original $\boldsymbol{v}_0$ and must also fulfill the equality constraints. This leads to the following constraint optimization problem

$$\left.\begin{aligned} \tilde{\boldsymbol{x}} &= \arg\min_{\boldsymbol{x}} ||\boldsymbol{x} - \boldsymbol{v}||, \\ \text{s.t.} \quad \boldsymbol{x}^{\mathrm{T}}\boldsymbol{\xi} &= W, \\ (\boldsymbol{x})_m &\geq 0, \ \forall m = 1, \ldots, M. \end{aligned}\right\} \tag{29}$$

Note, since the considerations do hold for all inner nodes, the "0" index has been dropped in (29). This optimization problem can also be interpreted as a projection of $\boldsymbol{v}$ onto the positive orthant. The resulting $\tilde{\boldsymbol{x}}$ fulfills both constraints (13b) and (5). The optimization problem (29) can be solved efficiently. An algorithm for solving (29) will be presented in the Appendix.

Having calculated the mutation in the root node, the pre-order walk through the tree can be done. As for the remaining inner nodes, condition (13d) changes to (5). As a result, (24) changes to

$$(\boldsymbol{x}_k + \tilde{\sigma}\boldsymbol{s}_k)^{\mathrm{T}}\boldsymbol{\xi}_k \overset{!}{=} \tilde{\boldsymbol{x}}_{\pi(k)}^{\mathrm{T}}\boldsymbol{\xi}_k. \tag{30}$$

Rearranging terms, one gets

$$\frac{(\tilde{\boldsymbol{x}}_{\pi(k)} - \boldsymbol{x}_k)^{\mathrm{T}}\boldsymbol{\xi}_k}{\tilde{\sigma}} \overset{!}{=} \sum_{m=1}^{M-1} (\boldsymbol{s}_k)_m(\boldsymbol{\xi}_k)_m + (\boldsymbol{s}_k)_M(\boldsymbol{\xi}_k)_M. \tag{31}$$

Solving for $(\boldsymbol{s}_k)_M$ yields

$$(\boldsymbol{s}_k)_M = \frac{1}{(\boldsymbol{\xi}_k)_M}\left(\frac{(\tilde{\boldsymbol{x}}_{\pi(k)} - \boldsymbol{x}_k)^{\mathrm{T}}\boldsymbol{\xi}_k}{\tilde{\sigma}} - \sum_{m=1}^{M-1} (\boldsymbol{s}_k)_m(\boldsymbol{\xi}_k)_m\right). \tag{32}$$

Choosing the first $M-1$ components $(\boldsymbol{s}_k)_m = (\boldsymbol{z}_k)_m$ and the Mth according to (32), one obtains the $\boldsymbol{s}_k$ vector

$$\boldsymbol{s}_k = \left((\boldsymbol{z}_k)_1, \ldots, (\boldsymbol{z}_k)_{M-1}, \frac{1}{(\boldsymbol{\xi}_k)_M}\left(\frac{(\tilde{\boldsymbol{x}}_{\pi(k)} - \boldsymbol{x}_k)^{\mathrm{T}}\boldsymbol{\xi}_k}{\tilde{\sigma}} - \sum_{m=1}^{M-1} (\boldsymbol{z}_k)_m(\boldsymbol{\xi}_k)_m\right)\right)^{\mathrm{T}}. \tag{33}$$

The raw offspring vector in the kth node is given by $\boldsymbol{v}_k$

$$\boldsymbol{v}_k = \boldsymbol{x}_k + \tilde{\sigma}\boldsymbol{s}_k. \tag{34}$$

Using (33) this yields

$$\begin{aligned} \boldsymbol{v}_k = \boldsymbol{x}_k + \Bigg(& (\tilde{\sigma}\boldsymbol{z}_k)_1, \ldots, (\tilde{\sigma}\boldsymbol{z}_k)_{M-1}, \\ & \frac{1}{(\boldsymbol{\xi}_k)_M}\left((\tilde{\boldsymbol{x}}_{\pi(k)} - \boldsymbol{x}_k)^{\mathrm{T}}\boldsymbol{\xi}_k - \sum_{m=1}^{M-1} (\tilde{\sigma}\boldsymbol{z}_k)_m(\boldsymbol{\xi}_k)_m\right)\Bigg)^{\mathrm{T}}. \end{aligned} \tag{35}$$

Again, the resulting $\boldsymbol{v}_k$ may violate the positivity constraint (13b). In such a case, the optimization problem (29) must be solved in order to find a feasible offspring $\tilde{\boldsymbol{x}}_k$.

Let us summarize the mutation process described by the Eqs. (24)–(35) in the pseudocode in Fig. 2. Line M1 implements (26) and (27). If there is a negative component in $\boldsymbol{v}_0$ (belonging to

```
Mutate($\mathbf{X}, \mathbf{Z}, \tilde{\sigma};\ \Xi, W_0$)

$\boldsymbol{v}_0 \leftarrow \boldsymbol{x}_0 + \tilde{\sigma}\left(\boldsymbol{z}_0^{\mathrm{T}},\ -\frac{1}{(\boldsymbol{\xi}_0)_M}\sum_{m=1}^{M-1}(\boldsymbol{z}_0)_m(\boldsymbol{\xi}_0)_m\right)^{\mathrm{T}}$      (M1)
If $\exists\, m = 1,\ldots,M:\ (\boldsymbol{v}_0)_m < 0$      (M2)
    Then $\tilde{\boldsymbol{x}}_0 \leftarrow \Pi(\boldsymbol{v}_0, \boldsymbol{\xi}_0, W_0)$      (M3)
    Else $\tilde{\boldsymbol{x}}_0 \leftarrow \boldsymbol{v}_0$      (M4)
For $k \leftarrow 1$ To $N - N_L$
    $\boldsymbol{v}_k \leftarrow \boldsymbol{x}_k + \tilde{\sigma}\left(\boldsymbol{z}_k^{\mathrm{T}},\ \frac{1}{(\boldsymbol{\xi}_k)_M}\left(\frac{(\tilde{\boldsymbol{x}}_{\pi(k)} - \boldsymbol{x}_k)^{\mathrm{T}}\boldsymbol{\xi}_k}{\tilde{\sigma}} - \sum_{m=1}^{M-1}(\boldsymbol{z}_k)_m(\boldsymbol{\xi}_k)_m\right)\right)^{\mathrm{T}}$      (M5)
    If $\exists\, m = 1,\ldots,M:\ (\boldsymbol{v}_k)_m < 0$      (M6)
        Then $\tilde{\boldsymbol{x}}_k \leftarrow \Pi(\boldsymbol{v}_k, \boldsymbol{\xi}_k, \tilde{\boldsymbol{x}}_{\pi(k)}^{\mathrm{T}}\boldsymbol{\xi}_k)$      (M7)
        Else $\tilde{\boldsymbol{x}}_k \leftarrow \boldsymbol{v}_k$      (M8)
End
```

Figure 2: Pseudocode of the mutation operator used in Line 7 of the mpCMSA-ES in Fig. 1. Note, $\mathbf{Z} = (\boldsymbol{z}_0, \ldots, \boldsymbol{z}_{N-N_L})$, $\mathbf{X} = (\boldsymbol{x}_0, \ldots, \boldsymbol{x}_{N-N_L})$, and $\Xi = (\boldsymbol{\xi}_0, \ldots, \boldsymbol{\xi}_{N-N_L})$.

the root node) the mutation must be repaired. This is done in M3. The function PositiveOrthantProjector $\Pi(\boldsymbol{v}, \boldsymbol{\xi}, W)$ returns the optimal solution to the constraint optimization problem (29): An admissible mutation must lie in the hyperplane $\boldsymbol{\xi}_0^{\mathrm{T}}\boldsymbol{x} = W_0$ and must not have any negative component. The PositiveOrthantProjector returns the minimal distance solution to this problem (therefore, it may be regarded as a projection). Having the mutation for the root node, one can now perform a pre-order walk through the remaining inner $N - N_L$ nodes. Line M5 implements (33) and (34). Again, if a mutation vector has a negative component it must be repaired in M7. The hyperplane parameter W_k is given by the wealth value dictated by the mutation vector of the predecessor node $\pi(k)$ and the asset prices at node k. The details of the PositiveOrthantProjector Π will be described in the Appendix.

3.3. Selection

As in each Evolution Strategy, selection is based on truncation of the ranked offspring population. The ranking has to respect the optimization goal (13a) *and* the VaR-constraint (13e). Therefore, the approach developed in [1] will be used: Due to the optimization problem setting, (13e) takes precedence over the f improvement (13a) as long as (13e) is violated. This suggest a lexicographic ordering to be applied to a fitness vector comprising $f(\mathbf{X})$ and $\nu(\mathbf{X})$. The latter is the *violation probability* function $\nu(\mathbf{X}) \in \{0, \ldots, 1\}$ adapted from [1]:

$$\nu(\mathbf{X}) := \begin{cases} 1 - \eta(\mathbf{X}), & \text{if } \eta(\mathbf{X}) < \alpha, \\ 0, & \text{otherwise.} \end{cases} \tag{36}$$

This function returns the sum of probabilities p_l for which $W_l < \kappa W_0$, $l \in \{N - N_L + 1, \ldots, N_L\}$, provided that this sum exceeds $(1-\alpha)$. Otherwise, it returns zero. The primary goal is to reach states $\mathbf{X}$ with zero violation probability $\nu(\mathbf{X})$. Therefore, an offspring $\tilde{\mathfrak{a}}_l$ is regarded to be better than another offspring $\tilde{\mathfrak{a}}_m$, i.e., $\tilde{\mathfrak{a}}_l \succ \tilde{\mathfrak{a}}_m$, if for $\nu(\tilde{\mathbf{X}}_m) > 0$ $\nu(\tilde{\mathbf{X}}_l) < \nu(\tilde{\mathbf{X}}_m)$. In the case of equality of the violation probabilities, the goal function values $f(\tilde{\mathbf{X}})$ are used in second place. Thus, the ordering relation reads

$$\tilde{\mathfrak{a}}_l \succ \tilde{\mathfrak{a}}_m \Leftrightarrow \Big(\nu(\tilde{\mathbf{X}}_l) < \nu(\tilde{\mathbf{X}}_m)\Big) \,\vee\, \Big[\Big(\nu(\tilde{\mathbf{X}}_l) = \nu(\tilde{\mathbf{X}}_m)\Big) \,\wedge\, \Big(f(\tilde{\mathbf{X}}_l) > f(\tilde{\mathbf{X}}_m)\Big)\Big]. \quad (37)$$

4. Experiments: Proof of Concept

In the experiments a $(30/30, 100)$-ES is used. Unlike the pseudocode, Fig. 1, the calculation of the transformation matrix $\mathbf{M} := \det(\mathbf{C})^{-\frac{1}{2K}}\sqrt{\mathbf{C}}$ in (L5) is stalled for a number of generations t_{stall}[4] given by $t_{\text{stall}} = \min\left(\lfloor\tau_c\rfloor, K/2\right)$. That is, t_{stall} does not exceed $K/2$ (at least every now and then there should be a update of $\mathbf{M}$ to allow for $\mathbf{C}$-matrix adaptation). In order to increase the probability of locating the optimizer, a restart version of the $(30/30, 100)$-ES (with random initialization) and $r = 10$ restarts has been used. The values for the termination criteria have been set to $\sigma_{stop} = 1, g_{stop} = 5000, G = 10$, and $\epsilon = 10^{-8}$. The seed capital is set to $W_0 = 10^6$.

In order to evaluate the performance of the mpCMSA-ES, reasonable scenario trees are needed. Such trees are generated from real time series data, e.g. data taken from the Dow Jones Index (DJI). In order to keep the problem computationally tractable, only a small number of assets $M = 5$ and small (aggregated) trees are considered in the investigations presented. Figure 3, left, shows the scenario tree used in the first investigations. It comprises four time periods obtained from a larger tree comprising nine periods and 190 inner nodes. Since the number of assets is $M = 5$, there are still $5 \times 15 = 75$ unknowns (located in the root node and the 14 inner nodes) to be determined. Note, the aim of the paper is concerned with algorithm design. Hence a rather small and computationally less demanding tree is sufficient for the proof of concept.

Demanding $\alpha = 1$ in (12) and (13), respectively, the system (13) becomes a linear stochastic program (see e.g. [2]). Feasible solutions will exist if κ is chosen sufficiently small in (11). In that case, the solutions produced with the mpCMSA-ES can be compared with the solution using linear programming (LP). Let $\hat{\kappa}$ be the value of κ above which no feasible solutions to the linear optimization problem do exist. Thus, one can easily check the solution quality of the mpCMSA-ES for $\kappa \le \hat{\kappa}$ using an LP solver. For the data provided, binary search yields $\hat{\kappa} \approx 1.01603$ ($f^* \approx 1{,}016{,}380$). Concerning the data given, there is only a small κ range below $\hat{\kappa}$ where better optimal LP solutions can be found $\hat{\kappa} \ge \kappa \ge \check{\kappa} \approx 0.975$.[5] Figure 4 compares the optimum values $f^*(\kappa)$ and $\mathbf{X}^*(\kappa)$ obtained by mpCMSA-ES with those of the LP solutions. As for the ES, the vicinity of $\hat{\kappa}$ is the hardest region since the ES has to find a solution in a very small domain of the search space where all inequalities must be fulfilled.

Comparing the differences in parameter space shows that for $\kappa \ge 0.99$ large differences in the solutions from mpCMSA-ES and LP exist. On the other hand, the differences in the objective are rather small. For example, for $\kappa = 1$ the difference in the objective value is about

[4]Self-evidently, there is also no need to recalculate these matrices for every single offspring anew.

[5]That is, choosing $\kappa < \check{\kappa}$ does not further increase the maximum value $f^*(\kappa)$. This is so because the domain defined by (13b)–(13d) is a proper subset of the solution domain of the inequality system $W_l \ge \kappa W_0$. Thus the LP solution is on the boundary of (13b)–(13d).

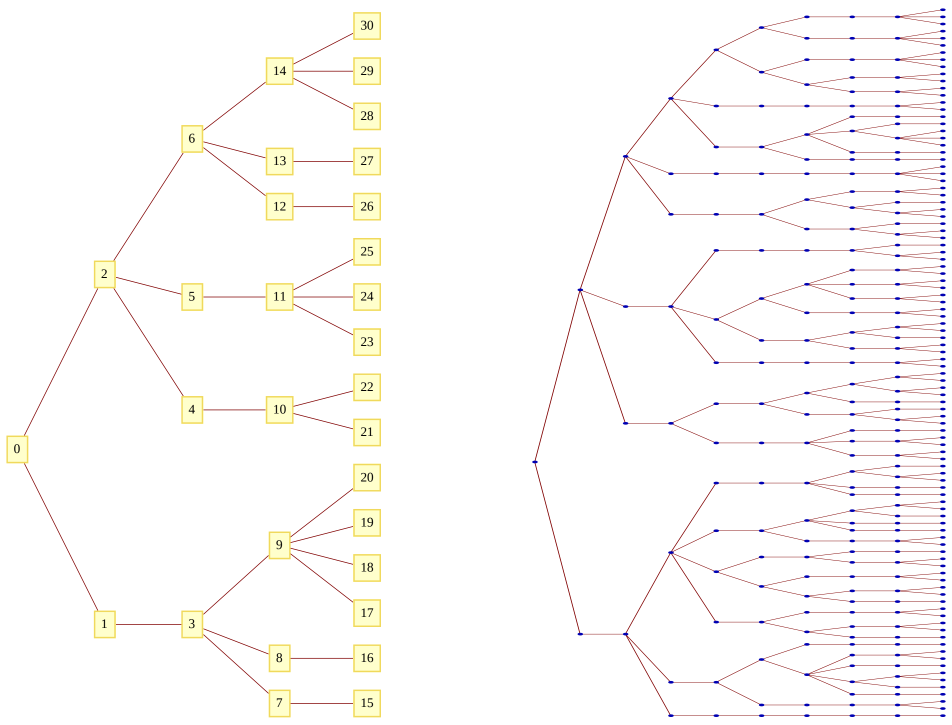

Figure 3: Left tree: Scenario tree having 16 leafs and 14 inner nodes. The asset weight vectors $\boldsymbol{x}$ in the root node and in the 14 inner nodes are to be optimized. Each of the 31 nodes has its own price vector $\boldsymbol{\xi}$. This scenario tree has been obtained by considering only the first 4 time steps from the tree on the rhs. Right tree: Scenario tree comprising 9 time steps. The latter has been provided by R. Kovacevic, University of Vienna. It comprises 100 leaf nodes and 190 inner nodes.

24.7 and the distance between the solutions is about $\|\mathbf{X}_{\mathrm{LP}} - \mathbf{X}_{\mathrm{mpCMSA-ES}}\| = 11518$. Closer examination of the solutions reveals that they differ by more than 1 in 20 of the 75 non-terminal portfolio weights, with 4 weights where either the LP solver or the mpCMSA uses non-zero weights. Comparing the time effort spend by the LP solver and the mpCMSA-ES, the former can solve the problem much faster. This was expected since the mpCMSA-ES is not tailored for LP problems.

The $(30/30, 100)$-mpCMSA-ES does not obtain feasible solutions up to $\hat{\kappa}$. One possibility to improve the solution quality is to increase the population size. However, experiments showed that mpCMSA-ES with smaller population sizes outperform versions with larger population sizes for some values of κ. Thus, for the remaining experiments in this section the restart version from [1] is used. In this restart scheme the overall budget of function evaluations is the same for each population size considered. The population sizes used range from $\lambda = 30$, where 64 independent runs are performed, up to $\lambda = 960$ with 2 independent runs.

Leaving the realm of linear feasibility $\kappa > \hat{\kappa}$, one enters the field of non-convex optimization where only α targets < 1 can be reached. As an example, we choose $\kappa = 1.02$ and $\alpha = 0.75$ since there exists a solution with a violation probability (36) of $\nu = 0$. Thus, one can compare with the solution calculated by the commercial optimization software LINGO ([24]

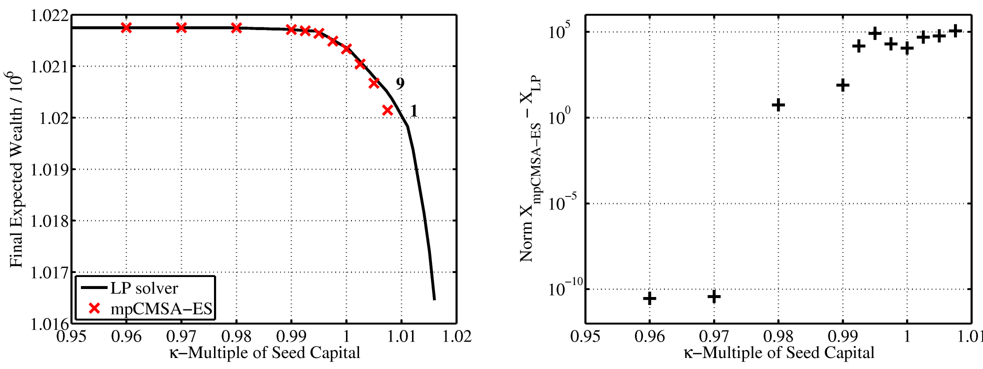

Figure 4: Comparison of the optimal solutions obtained by a $(30/30, 100)$-mpCMSA-ES with those of an LP solver in objective space (top plot) and parameter space (bottom plot) in the feasible $\kappa \leq \hat{\kappa}$ region. Note, as κ decreases, a saturation occurs at a certain $\kappa < \hat{\kappa}$ below which κ does not influence the optimum solution any longer. The numbers next to the markers in the left plot indicate the number of feasible solutions obtained in 10 independent restarts if not all solutions returned were feasible.

Version 12.0). LINGO returns only results if these represent feasible solutions. In contrast, the mpCMSA-ES returns also the best η, Eq. (11), reached so far even when (12) is not fulfilled. Thus, one gets the information as to what can be reached at least even if there is no feasible solution. Figure 5 shows the dynamics of the evolution process. The maximal f value is about $1.0202 \cdot 10^6$ and the number of inequalities not fulfilled is 3. This is in agreement with the result of LINGO's global optimization solver. The deviation in the maximal objective value is about 0.05%. In the top right graph of Fig. 5 the VaR violation

$$\rho := 1 - \eta(\mathbf{X}) \tag{38}$$

is displayed. It reaches a value of $\rho = 0.23$ for the best solution while the target is $\alpha = 0.75$.

Due to the 15 equality constraints in the 14 inner node and the root node, the actual dimensionality of the covariance matrix $\mathbf{C}$ in the mpCMSA-ES is $K \times K$ with $K = 60$ $(M = 5)$ obtained by (14). The lower left picture in Fig. 5 displays the evolution of selected eigenvalues of the stalled $\mathbf{C}$ matrix including the largest (top curve) and the smallest (bottom curve) eigenvalue. As one can infer from these dynamics and the corresponding f and ρ dynamics, using non-isotropic Gaussians is needed to efficiently reach the vicinity of the optimum.

Comparing mpCMSA-ES runs with and without covariance adaptation showed that the latter setup is always inferior. As example, using the same problem and algorithmic setup as used for Fig. 5 the mpCMSA-ES without covariance matrix adaptation did not yield a feasible solution.

In order to compare the solution quality of the mpCMSA-ES with LINGO in more detail, the dependency of the VaR violation ρ on κ has been investigated (to this end, $\alpha = 1$ has been used). The mpCMSA-ES applying restarts with increasing population size has been used. The results are displayed in the left graph of Fig 6. Being based on these results a confidence level of $\alpha = 0.85$ has been chosen to investigate the dependence of the f maximum on the choice of κ. The graph on the right in Fig 6 displays the result.

Comparing the performance of LINGO and mpCMSA-ES one observes that both achieve more or less the same solution quality except for $\kappa \geq 1.015$. For $\kappa > 1.1015$ only LINGO finds

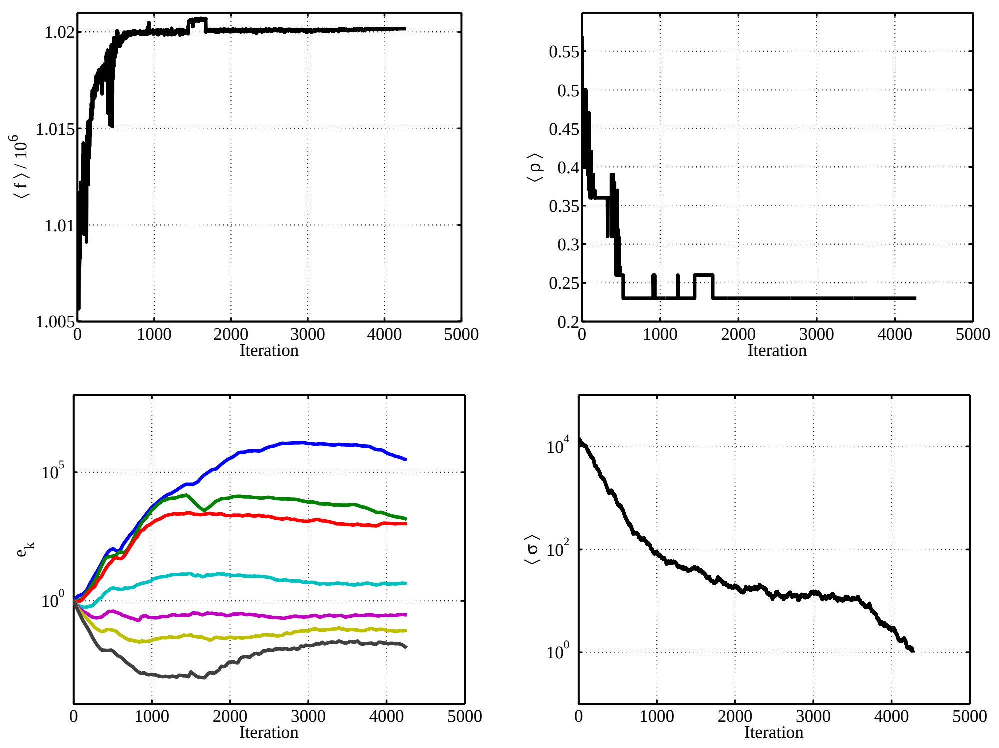

Figure 5: Dynamics of the $(30/30, 100)$-mpCMSA-ES on a problem instance of the left tree displayed in Fig. 3. The mpCMSA-ES terminated after reaching the stop criterion $\sigma_{\text{stop}} = 1$ at about 4276 generations. It reaches a VaR violation of $\rho = 0.23$ (top right graph), thus, fulfilling the $\alpha = 0.75$ condition in (12). The bottom left graph shows the dynamics of some of the 60 eigenvalues e_k of the (stalled) $\mathbf{C}$ matrix being the $k =$ 1st (largest), $k =$ 2nd, $k =$ 3rd, 15., 30., 45., 60. (smallest) eigenvalue (from top to bottom).

a feasible solution for the considered κ-values. If both find a feasible solution, the VaR violation ρ of these solutions is identical in 9 out of 11 considered κ-values. The use of mpCMSA-ES with different population sizes is validated, since different population sizes provide the best solutions over the κ range considered. However, for the κ values considered the worst feasible solution is at least $0.9947 f_{\max}$. A drawback of the mpCMSA-ES is the high cost to obtain a good solution. Whether different restart schemes with simultaneous population size adaptation improve the overall behavior could be an area for future research.

In this section we have shown that the mpCMSA-ES is able to find solutions to the multi-periodic optimization problem (13). However, this problem is dominated by linear equality and linear inequality constraints. Therefore, nonlinear approximation algorithms that linearly approximate the only nonlinearity (13e) piecewise are expected to perform well on such problems resulting in a sequence of linear programs. This is probably the reason why LINGO performs well on this problem class. In order to make the problem more nonlinear and - of course - more realistic, one has to consider (13) with transaction costs.

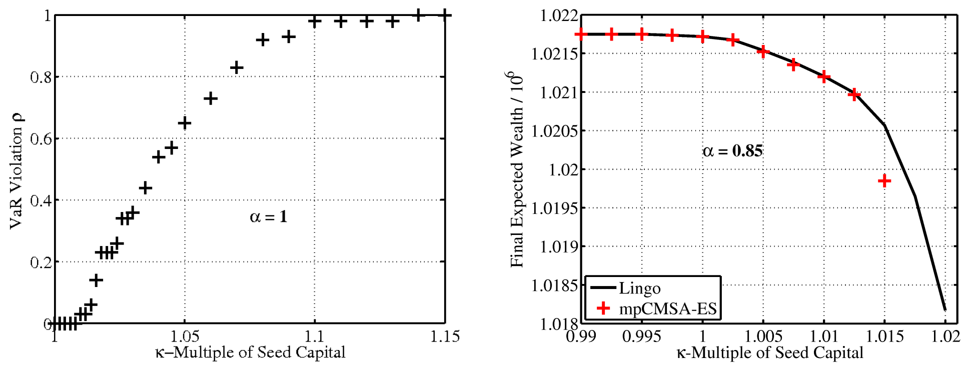

Figure 6: *Left plot*: VaR violation ρ depending on the choice of κ for $\alpha = 1$. *Right plot*: Comparison of the $f_{\max}$ distributions for LINGO (curve) and mpCMSA-ES (+) as function of κ for $\alpha = 0.85$. All solutions shown are feasible.

5. Including transaction costs

5.1. How to Ensure the Nonlinear Equality Constraints

The transaction costs described by the BS functions in (6) and (7) are nonlinear functions of the decision vectors. While it is still possible to perform the mutations by traversing the tree, one has to ensure nonlinear equality constraints in each inner node. Depending on BS, this can be a very demanding problem. This holds especially if in model (7) the fixed costs Mc_f already exceed the wealth $W_{\pi(k)} = \boldsymbol{x}^{\mathrm{T}}_{\pi(k)}\boldsymbol{\xi}_k$ in node k. In such cases, buy or sell actions are only allowed for a subset of the assets in node k. Leaving aside this rather hard case and assuming BS small compared to $W_{\pi(k)} = \boldsymbol{x}^{\mathrm{T}}_{\pi(k)}\boldsymbol{\xi}_k$, fixed point iteration may be used to solve (13c). That is, given $W_\pi = \boldsymbol{x}^{\mathrm{T}}_\pi\boldsymbol{\xi}$, BS is considered as a (small) perturbation of W_π leading from (13c) to the equation (dropping the node index k)

$$\boldsymbol{x}^{\mathrm{T}}\boldsymbol{\xi} = \boldsymbol{x}^{\mathrm{T}}_\pi\boldsymbol{\xi} - \mathrm{BS}(\boldsymbol{x}_\pi, \boldsymbol{x}, \boldsymbol{\xi}) =: \Delta W. \tag{39}$$

Equation (39) would be linear w.r.t. $\boldsymbol{x}$ if its rhs would not depend on $\boldsymbol{x}$. This can be accomplished by interpreting (39) as an iterative scheme where $\boldsymbol{x}$ is $\boldsymbol{x} = \boldsymbol{x}^{(t)}$ on the rhs being the $\boldsymbol{x}$ at iteration t and $\boldsymbol{x} = \boldsymbol{x}^{(t+1)}$ remains to be determined according to

$$\boldsymbol{\xi}^{\mathrm{T}}\boldsymbol{x}^{(t+1)} = \boldsymbol{x}^{\mathrm{T}}_\pi\boldsymbol{\xi} - \mathrm{BS}(\boldsymbol{x}_\pi, \boldsymbol{x}^{(t)}, \boldsymbol{\xi}) = \Delta W^{(t)}. \tag{40}$$

Provided that $\Delta W^{(t)} > 0$, (40) can be solved by applying the PositiveOrthantProjector Π yielding

$$\boldsymbol{x}^{(t+1)} = \Pi(\boldsymbol{x}^{(t)}, \boldsymbol{\xi}, \Delta W^{(t)}). \tag{41}$$

Using a start vector $\boldsymbol{x}^{(0)}$ the projection (41) should be a contractive mapping, thus approaching the fixed point of the mapping for $t \to \infty$. However, to this end, $\Delta W^{(t)} > 0$ must be ensured. Provided that the fixed costs Mc_f are less than the wealth of the node W_π this can be ensured by an additional linear scaling of the $\boldsymbol{x}^{(t)}$ vector. Such scalings are admissible since the $\boldsymbol{x}$ finally realized in a specific node is just an offspring in the evolutionary process. That is, the result of this scaling and the iterative mapping (41) can be regarded as just another (repaired) mutation applied to the parental state in (L7) in Fig. 1.

If the $\Delta W > 0$ condition is not fulfilled in (41), $\boldsymbol{x}^{(t)}$ must be scaled accordingly

$$\check{\boldsymbol{x}} = \boldsymbol{x}_\pi + r(\boldsymbol{x} - \boldsymbol{x}_\pi). \tag{42}$$

Here we have dropped the iteration counter for brevity. By choosing $r \in [0, 1]$ one can tune $\check{\boldsymbol{x}}$ in such a manner that $\Delta W > 0$ is fulfilled for $\boldsymbol{x}^{(t)} = \check{\boldsymbol{x}}$. In order to prove this assertion, consider BS_2 and the rhs of (40). One has to choose $\check{\boldsymbol{x}}$ in such a way that $W_\pi > \mathrm{BS}_2(\boldsymbol{x}_\pi, \check{\boldsymbol{x}}, \boldsymbol{\xi})$. Using (7) together with (42) yields

$$\begin{aligned} W_\pi > c_f \sum_{m=1}^{M} \overline{\delta}(r((\boldsymbol{x}_\pi)_m - (\boldsymbol{x})_m)) + c_b \sum_{m=1}^{M} \Theta(r((\boldsymbol{x})_m - (\boldsymbol{x}_\pi)_m)) r^*_m \\ + c_s \sum_{m=1}^{M} \Theta(r((\boldsymbol{x}_\pi)_m - (\boldsymbol{x})_m)) r^*_m \end{aligned} \tag{43}$$

where $r^*_m := |r((\boldsymbol{x}_\pi)_m - (\boldsymbol{x})_m)| (\boldsymbol{\xi})_m$. Since for $r \neq 0 : \overline{\delta}(rz) = \overline{\delta}(z)$ and for $r \geq 0 : \Theta(rz) = \Theta(z)$ one obtains

$$\begin{aligned} W_\pi - c_f \sum_{m=1}^{M} \overline{\delta}((\boldsymbol{x}_\pi)_m - (\boldsymbol{x})_m) > c_b r \sum_{m=1}^{M} \Theta((\boldsymbol{x})_m - (\boldsymbol{x}_\pi)_m) |(\boldsymbol{x}_\pi)_m - (\boldsymbol{x})_m| \, (\boldsymbol{\xi})_m \\ + c_s r \sum_{m=1}^{M} \Theta((\boldsymbol{x}_\pi)_m - (\boldsymbol{x})_m) |(\boldsymbol{x}_\pi)_m - (\boldsymbol{x})_m| \, (\boldsymbol{\xi})_m. \end{aligned} \tag{44}$$

Resolving for r yields

$$r < \frac{W_\pi - c_f \sum_{m=1}^{M} \overline{\delta}((\boldsymbol{x}_\pi)_m - (\boldsymbol{x})_m)}{\boldsymbol{D}}, \tag{45}$$

where

$$\boldsymbol{D} := \sum_{m=1}^{M} \left[c_b \Theta((\boldsymbol{x})_m - (\boldsymbol{x}_\pi)_m) + c_s \Theta((\boldsymbol{x}_\pi)_m - (\boldsymbol{x})_m) \right] |(\boldsymbol{x}_\pi)_m - (\boldsymbol{x})_m| \, (\boldsymbol{\xi})_m. \tag{46}$$

Therefore, choosing

$$r = \beta \, \frac{W_\pi - c_f \sum_{m=1}^{M} \overline{\delta}((\boldsymbol{x}_\pi)_m - (\boldsymbol{x})_m)}{\boldsymbol{D}}, \tag{47}$$

where $\beta \in (0, 1)$ ensures $\Delta W > 0$. Using BS_2 (7), r can be expressed as

$$r = \beta \, \frac{W_\pi - \mathrm{BS}_2(\boldsymbol{x}_\pi, \boldsymbol{x}, \boldsymbol{\xi})|_{c_b = c_s = 0}}{\mathrm{BS}_2(\boldsymbol{x}_\pi, \boldsymbol{x}, \boldsymbol{\xi})|_{c_f = 0}} \tag{48}$$

and the rescaled $\boldsymbol{x}$ becomes using (47)

$$\check{\boldsymbol{x}} = \boldsymbol{x}_\pi + \beta \, \frac{W_\pi - \mathrm{BS}_2(\boldsymbol{x}_\pi, \boldsymbol{x}, \boldsymbol{\xi})|_{c_b = c_s = 0}}{\mathrm{BS}_2(\boldsymbol{x}_\pi, \boldsymbol{x}, \boldsymbol{\xi})|_{c_f = 0}} \, (\boldsymbol{x} - \boldsymbol{x}_\pi). \tag{49}$$

It remains to calculate the resulting new ΔW. Substituting (49) in (39) using BS_2, one gets

$$\begin{aligned}
\Delta W &= W_\pi - \mathrm{BS}_2(\boldsymbol{x}_\pi, \check{\boldsymbol{x}}, \boldsymbol{\xi}) \\
&= W_\pi - \mathrm{BS}_2(\boldsymbol{x}_\pi, \boldsymbol{x}_\pi + r(\boldsymbol{x} - \boldsymbol{x}_\pi), \boldsymbol{\xi}) \\
&= W_\pi - \mathrm{BS}_2(\boldsymbol{x}_\pi, \boldsymbol{x}, \boldsymbol{\xi})|_{c_b=c_s=0} - r\mathrm{BS}_2(\boldsymbol{x}_\pi, \boldsymbol{x}, \boldsymbol{\xi})|_{c_f=0} \\
&= W_\pi - \mathrm{BS}_2(\boldsymbol{x}_\pi, \boldsymbol{x}, \boldsymbol{\xi})|_{c_b=c_s=0} \\
&\qquad - \beta \frac{W_\pi - \mathrm{BS}_2(\boldsymbol{x}_\pi, \boldsymbol{x}, \boldsymbol{\xi})|_{c_b=c_s=0}}{\mathrm{BS}_2(\boldsymbol{x}_\pi, \boldsymbol{x}, \boldsymbol{\xi})|_{c_f=0}} \mathrm{BS}_2(\boldsymbol{x}_\pi, \boldsymbol{x}, \boldsymbol{\xi})|_{c_f=0} \\
&= (1-\beta)\left[W_\pi - \mathrm{BS}_2(\boldsymbol{x}_\pi, \boldsymbol{x}, \boldsymbol{\xi})|_{c_b=c_s=0}\right].
\end{aligned} \tag{50}$$

As one can see, β reduces linearly the amount of wealth ΔW that can be redistributed by $\boldsymbol{x}$. As a reasonable choice, $\beta = 1/2$ may be considered as a value to start with. Also note that the BS_1 function (6) is included in the formulae (49) and (50) as a special case of BS_2 for $c_f \equiv 0$ and $c_b = c_s = c$. Putting things together, Fig. 7 displays the resulting algorithm to obtain an offspring $\boldsymbol{x}$ fulfilling the nonlinear equality constraint (13c) in each inner node k. This algorithm has two

```
EnsureBS_2Constraints($\boldsymbol{x}_\pi, \boldsymbol{v}, \boldsymbol{\xi}$)

$W_\pi \leftarrow \boldsymbol{x}_\pi^{\mathrm{T}}\boldsymbol{\xi}$                                  (E1)
If $W_\pi \le c_f$ Return $\boldsymbol{x}_\pi$                          (E2)
$t \leftarrow 0$                                             (E3)
$\boldsymbol{x}^{(t)} \leftarrow \boldsymbol{v}$                                       (E4)
Do
  If $t \ge t_{max}$ Return $\boldsymbol{x}_\pi$                        (E5)
  $\Delta W \leftarrow W_\pi - \mathrm{BS}_2(\boldsymbol{x}_\pi, \boldsymbol{x}^{(t)}, \boldsymbol{\xi})$              (E6)
  If $\Delta W \le 0$                                        (E7)
    $\boldsymbol{x}^{(t)} \leftarrow \boldsymbol{x}_\pi + \beta \frac{W_\pi - \mathrm{BS}_2(\boldsymbol{x}_\pi, \boldsymbol{x}^{(t)}, \boldsymbol{\xi})|_{c_b=c_s=0}}{\mathrm{BS}_2(\boldsymbol{x}_\pi, \boldsymbol{x}^{(t)}, \boldsymbol{\xi})|_{c_f=0}} (\boldsymbol{x}^{(t)} - \boldsymbol{x}_\pi)$   (E8)
    $\Delta W \leftarrow (1-\beta)\left[W_\pi - \mathrm{BS}_2(\boldsymbol{x}_\pi, \boldsymbol{x}^{(t)}, \boldsymbol{\xi})|_{c_b=c_s=0}\right]$   (E9)
  End
  $\boldsymbol{x}^{(t+1)} \leftarrow \Pi(\boldsymbol{x}^{(t)}, \boldsymbol{\xi}, \Delta W)$                  (E10)
  If $\left| \boldsymbol{\xi}^{\mathrm{T}}\boldsymbol{x}^{(t+1)} + \mathrm{BS}_2(\boldsymbol{x}_\pi, \boldsymbol{x}^{(t+1)}, \boldsymbol{\xi}) - W_\pi \right| < \epsilon$ Return $\boldsymbol{x}^{(t+1)}$   (E11)
  $\boldsymbol{x}^{(t)} \leftarrow \boldsymbol{x}^{(t+1)}$                                (E12)
  $t \leftarrow t + 1$                                       (E13)
End
```

Figure 7: Pseudocode for fulfilling the nonlinear equality constraint in a node parameterized by the pre-node $\boldsymbol{x}$, the raw offspring value $\boldsymbol{v}$ proposed by the mutation process, and the assets price vector $\boldsymbol{\xi}$ in that node. This algorithm returns always an $\boldsymbol{x}$ vector fulfilling (13c). However, it simply returns the pre-node $\boldsymbol{x}$ value $\boldsymbol{x}_\pi$ if the fixed price for a transaction already exceeds the wealth of the node or if the algorithm does not converge in a predefined number of iteration steps t_{max}. A reasonable choice for β is $\beta = 1/2$. The precision by which the equality constraint (13c) is to be fulfilled may be chosen as $\epsilon = 10^{-8}$.

abnormally terminations in (E2) and (E5). In both cases it is not able to find an $\boldsymbol{x}_k \neq \boldsymbol{x}_{\pi(k)}$ fulfilling the constraint (13c). This is a fallback solution that is not really desirable, a more refined algorithm should also deal with such situations. However, this is beyond the scope of this paper.

The formal parameter vector $\boldsymbol{v}$ is the nodal raw individual vector provided by the raw parent initialization step (L1, Fig. 1) or the raw mutation process performed in the mutation pseudocode, Fig. 2, in Lines (M1) and (M5), respectively (see also Pseudocode, Fig. 8, below). Fixed point iteration is performed in the Do-loop. In order to use the PositiveOrthantProjector, the BuySell cost must not exceed the wealth $W_\pi = \boldsymbol{x}_\pi^{\mathrm{T}}\boldsymbol{\xi}$ determined by the pre-node $\boldsymbol{x}$. If this is not fulfilled, the current iterate $\boldsymbol{x}^{(t)}$ must be changed towards $\boldsymbol{x}_\pi$ by means of Eqs. (42), (48), (49). This is performed in (E8). The respective ΔW change is performed in (E9) implementing (50). If the condition $\Delta W > 0$ is fulfilled, the projection of $\boldsymbol{x}^{(t)}$ fulfilling (A.1) takes place in (E10). If the resulting $\boldsymbol{x} = \boldsymbol{x}^{(t+1)}$ does fulfill the equality constraint (13c) sufficiently well, it can be assumed that $\boldsymbol{x}$ is close to the fixed point and the fixed point iteration is stopped returning $\boldsymbol{x}^{(t+1)}$ in (E11) as a solution to (13c). Note, if $\mathrm{BS}_2 \equiv 0$ then (E7) is never fulfilled (thus, there is no danger of zero division) and the iteration stops after the first projection (E10) since the condition in (E11) is immediately fulfilled. That is, the special case of zero costs considered in Section 3 is covered as well.

5.2. Adapting Initialization, Mutation, and Recombination

The initialization of the root node can be done according to Eq. (22). Changes are necessary for Eq. (23) concerning the successor nodes. The resulting $\boldsymbol{x}_k$ using (23) will usually not fulfill (13c). Therefore, the algorithm in Fig. 7 must be used in conjunction with (23) to get a feasible parent individual

$$\boldsymbol{x}_k = \text{EnsureBS}_2\text{Constraints}\left(\boldsymbol{x}_{\pi(k)}, \frac{\boldsymbol{x}_{\pi(k)}^{\mathrm{T}}\boldsymbol{\xi}_k}{\boldsymbol{u}_k^{\mathrm{T}}\boldsymbol{\xi}_k}\boldsymbol{u}_k, \boldsymbol{\xi}_k\right). \tag{51}$$

The mutation of the parental state is rather similar to the algorithm in Fig. 2 with the exception that (M6)–(M8) must be replaced to ensure the nonlinear constraints (13c). The resulting algorithm is given in Fig. 8. It deviates from the original one only in Line (M6*).

Due to the nonlinearity of the equality constraint (13c), the recombination operation in (L16) of the mpCMSA-ES does *not* guarantee the production of feasible parent individuals. There is no generic way to ensure feasibility after recombination. This holds also for other types of recombination. There are three options to replace (L16) in Fig 1:

(a) Abandoning recombination using $\tilde{\mathbf{X}}$ of the $\tilde{\mathfrak{a}}_{1;\lambda}$ individual as parent of the next generation, i.e.,

$$\mathbf{X}^{(g)} \leftarrow \tilde{\mathbf{X}}_{1;\lambda}. \tag{52}$$

(b) Repairing the recombinant produced in (L16) similar to the repair done for the offspring in Eq. (51), i.e.,

$$\forall k = 1, \ldots, N - N_L : \boldsymbol{x}_k^{(g)} \leftarrow \text{EnsureBS}_2\text{Constraints}\left(\boldsymbol{x}_{\pi(k)}, \langle\tilde{\boldsymbol{x}}_k\rangle, \boldsymbol{\xi}_k\right). \tag{53}$$

(c) Ignoring the violation of (13c) in the parental state. In that case, only offspring are guaranteed to be feasible and only the best offspring should be returned after termination of the mpCMSA-ES.

Running experiments with the different setups did not reveal any significant differences in the results. Therefore, option (b) was chosen, however, the recombinant is replaced if the best offspring is superior.

```
                    Mutate*(X, Z, σ̃; Ξ, W_0)
$\boldsymbol{v}_0 \leftarrow \boldsymbol{x}_0 + \tilde\sigma \left( \boldsymbol{z}_0^{\mathrm{T}},\ -\frac{1}{(\boldsymbol{\xi}_0)_M} \sum_{m=1}^{M-1} (\boldsymbol{z}_0)_m (\boldsymbol{\xi}_0)_m \right)^{\mathrm{T}}$        (M1)
If $\exists\, m = 1, \ldots, M :\ (\boldsymbol{v}_0)_m < 0$        (M2)
    Then $\tilde{\boldsymbol{x}}_0 \leftarrow \Pi(\boldsymbol{v}_0, \boldsymbol{\xi}_0, W_0)$        (M3)
    Else $\tilde{\boldsymbol{x}}_0 \leftarrow \boldsymbol{v}_0$        (M4)
For $k \leftarrow 1$ To $N - N_L$
    $\boldsymbol{v}_k \leftarrow \boldsymbol{x}_k + \tilde\sigma \left( \boldsymbol{z}_k^{\mathrm{T}},\ \frac{1}{(\boldsymbol{\xi}_k)_M} \left( \frac{(\tilde{\boldsymbol{x}}_{\pi(k)} - \boldsymbol{x}_k)^{\mathrm{T}} \boldsymbol{\xi}_k}{\tilde\sigma} - \sum_{m=1}^{M-1} (\boldsymbol{z}_k)_m (\boldsymbol{\xi}_k)_m \right) \right)^{\mathrm{T}}$        (M5)
    $\tilde{\boldsymbol{x}}_k \leftarrow \mathrm{EnsureBS_2Constraints}\left(\boldsymbol{x}_{\pi(k)}, \boldsymbol{v}_k, \boldsymbol{\xi}_k\right)$        (M6*)
End
```

Figure 8: Pseudocode of the mutation operator that respects the non-linear equilibrium condition (13c). This algorithm is to be used in Line 7 of the mpCMSA-ES in Fig. 1. Note, $\mathbf{Z} = (\boldsymbol{z}_0, \ldots, \boldsymbol{z}_{N-N_L})$, $\mathbf{X} = (\boldsymbol{x}_0, \ldots, \boldsymbol{x}_{N-N_L})$, and $\boldsymbol{\Xi} = (\boldsymbol{\xi}_0, \ldots, \boldsymbol{\xi}_{N-N_L})$.

5.3. Experiments with non-proportional Transaction Costs

In the first experiments we use again a $(30/30, 100)$-mpCMSA-ES, i.e., recombination of the selected parents is performed. This implies the application of option (c) or (b) discussed at the end of the last Section 5.2. For the experiments we decided to repair the recombinant according to Eq. (53). The cost model (7) uses $c_f = 50$ for the fixed cost for each asset and $c_b = 0.002$ and $c_s = 0.003$ for buy and sell actions[6], respectively. Figure 9 shows the dynamics of a typical run. As one can see, given the parameters $\kappa = 1$ and $\alpha = 0.85$, the ES reaches a better VaR violation as demanded ($\rho = 0.15$ would suffice). The f value obtained by the ES is $f = 1.0119 \cdot 10^6$. Applying LINGO to the same problem yields slightly worse results $f = 1.010 \cdot 10^6$ with $\rho = 0.12$.

In order to compare the solution quality of the mpCMSA-ES with LINGO in more detail, the dependency of the relative violation count ρ on κ has been investigated (to this end, $\alpha = 1$ has been used). The results are displayed in the left graph of Fig 10. Being based on these results a confidence level of $\alpha = 0.8$ has been chosen to investigate the dependence of the f maximum on the choice of κ. The right graph in Fig 10 displays the result. For the graphs in Fig. 10 the restart version of the mpCMSA-ES has been used. The runs with LINGO were terminated after at most 24 hours. In the rhs of Fig. 10 one observes that the mpCMSA-ES outperforms LINGO for several values of κ. For $\kappa \geq 1$ not all LINGO instances performed yielded a feasible solution. For some values of κ, the mpCMSA-ES required smaller initial mutation strengths to converge, especially for $\lambda = 960$. This shows that the achievable solution quality for (some) κ values does depends on the initial mutation strength. As before, the best solutions from mpCMSA-ES occur at different population sizes, however, for the majority of these the maximal allowable number of constraint violations is exploited.

[6]Our initial values were 10 times higher, however, this resulted in solutions were no transactions were executed.

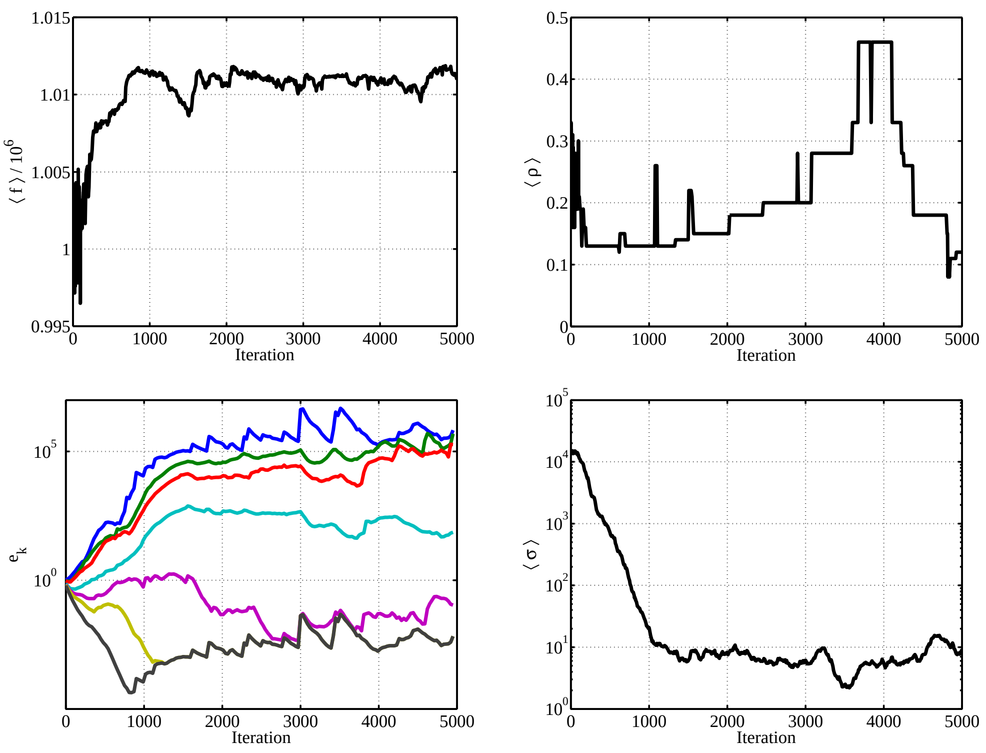

Figure 9: Dynamics of the $(30/30, 100)$-mpCMSA-ES on a problem instance of the left tree displayed in Fig. 3 with nonlinear cost constraints parameterized as $c_f = 50$, $c_b = 0.002$, and $c_s = 0.003$. The mpCMSA-ES terminated after reaching the stop criterion of 5000 generations. It reaches a VaR violation of $\rho = 0.11$ for the best solution (top right graph), thus, fulfilling the $\alpha = 0.85$ condition in (12). The bottom left graph shows the dynamics of some of the 60 eigenvalues e_k of the (stalled) $\mathbf{C}$ matrix being the $k = 1$st (largest), 15., 30., 45., 60. (smallest) eigenvalue (from top to bottom).

6. Summary and Outlook

Multi-periodic portfolio optimization poses non-convex optimization problems the solutions of which are computationally demanding tasks. In this paper an approach using CMSA Evolution Strategies has been presented that offers a solution approach. Up to now, the optimization problem class was not a typical application domain for Evolutionary Algorithms. This is understandable because the overwhelming number of constraints in such optimization problems are of linear type. Therefore, solution approaches based on linear programming (LP) are the appropriate means that promise success even if there are some additional nonlinear constraints. The latter can be linearized and thus be incorporated in the LP framework. This raises the question why we consider evolutionary approaches at all in this field. There are at least three aspects that should be taken into account:

- The design of ESs for constraint optimization is rather underdeveloped. Using the standard techniques usually proposed for evolutionary algorithms (for an overview, see e.g. [25,

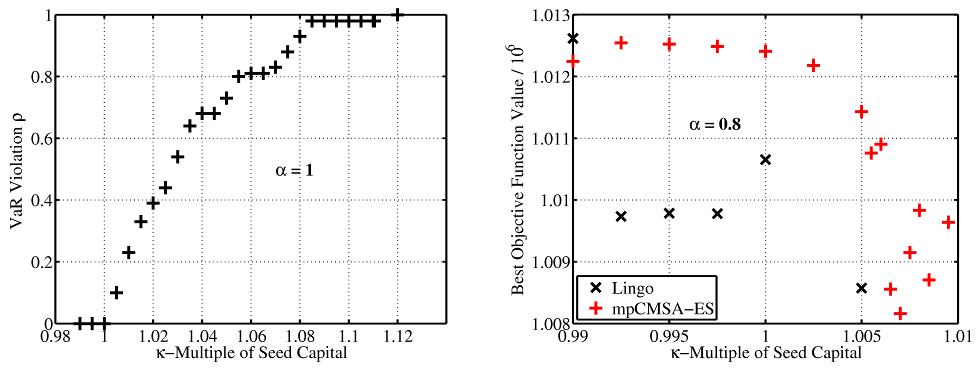

Figure 10: *Left plot*: Relative violation counts depending on the choice of κ for $\alpha = 1$. *Right plot*: Comparison of the $f_{\max}$ distributions for LINGO ($\times$) and mpCMSA-ES ($+$) as function of κ for $\alpha = 0.8$. All solutions shown are feasible.

26]) do not work well if the optimum is located on the boundary of feasibility. One often observes premature convergence due to premature step-size convergence [1]. Therefore, there is a certain demand for constraint handling techniques for ES which allow for a robust evolution on such boundaries.

- Constraints organized w.r.t. a tree structure as it appears in stochastic programming have not been treated with EAs so far (see, however, [27]).

- Evolution on trees does not only appear in financial optimization problems, it also appears in other fields where constraints can be represented by dependency trees. The method developed allows for the treatment of *nonlinear* balance equations in plant and process engineering.

The results reported in this paper are promising. However, one has to clearly state that the application domain is on strong nonlinear constraints and moderate search space dimensionalities. We would not recommend using the technique developed for simple linear stochastic programming problems.

While we have provided a proof of concept for the new-developed mpCMSA-ES, additional research is needed to compare the solutions obtained by evolutionary search with those obtained by LP and other techniques. Especially the question of the solution robustness should be investigated. Since the optimization problems considered are derived from scenario trees, the model to be optimized bears considerable model uncertainties. As a result, the optimizer obtained are also not very reliable. Therefore, it does not make sense to put too much effort in finding the global optimum, instead we are searching for solutions that are rather insensitive, i.e. robust, w.r.t. model uncertainties. Evolutionary search methods [28] might be a means to find such robust solutions.

Another aspect for future investigations are methods for decreasing the complexity, and therefore the runtime, of the mpCMSA-ES. One possibility is to learn only a set of the eigenvalues for the covariance matrix. This will speed up the learning of the covariance matrix, albeit with an inherent approximation error. Such methods will also be useful for "standard" applications of CMSA-ES, and possibly CMA-ES, in large search space dimensionalities.

Acknowledgments

The authors thank Raimund Kovacevic for providing the problem formulation and his solution to the optimization problem using the commercial global optimization solver Lingo. This work was supported by the Austrian Science Fund (FWF) under grants L532-N18 and P22649-N23.

Appendix A. How to Project onto the Positive Orthant

In order to repair infeasible solutions $\boldsymbol{v}$, the optimization problem (29)

$$\left.\begin{aligned} \tilde{\boldsymbol{x}} &= \arg\min_{\boldsymbol{x}} ||\boldsymbol{x}-\boldsymbol{v}||^2, \\ \text{s.t.} \quad & \tilde{\boldsymbol{x}}^{\mathrm{T}}\boldsymbol{\xi} = W, \\ & (\tilde{\boldsymbol{x}})_m \geq 0, \ \forall m = 1, \ldots, M. \end{aligned}\right\} \tag{A.1}$$

must be solved. Note, for sake of simplicity, the squared length is used in the first line of (A.1). While there is already an algorithm for projecting onto the probabilistic simplex under the constraint $\sum_{m=1}^{M} x_m = 1$ [29], no algorithm has been found for (A.1) in literature. Therefore, we will provide a derivation for an efficient algorithm that performs the optimization in (at most) $\mathcal{O}(M^2)$.

Let us first assume that (A.1) is without the inequality constraint $x_m > 0$. In that case one can immediately calculate the optimal $\boldsymbol{x}$ vector using Lagrange's approach. Starting from

$$L(\boldsymbol{x}, \eta) := \frac{1}{2}\|\boldsymbol{x}-\boldsymbol{v}\|^2 + \eta(\boldsymbol{\xi}^{\mathrm{T}}\boldsymbol{x} - W), \tag{A.2}$$

calculating the derivatives w.r.t. x_m and η and equating to zero yields

$$x_m - v_m + \eta\xi_m = 0 \tag{A.3}$$

and

$$\boldsymbol{\xi}^{\mathrm{T}}\boldsymbol{x} - W = 0. \tag{A.4}$$

Solving for x_m in (A.3) yields

$$x_m = v_m - \eta\xi_m. \tag{A.5}$$

Inserting this result in (A.4), one obtains

$$\sum_{m=1}^{M} \xi_m v_m - \eta \sum_{m=1}^{M} \xi_m^2 - W = 0 \tag{A.6}$$

and after resolving for η

$$\eta = \frac{\sum_{m=1}^{M} \xi_m v_m - W}{\sum_{m=1}^{M} \xi_m^2} = \frac{\boldsymbol{\xi}^{\mathrm{T}}\boldsymbol{v} - W}{\|\boldsymbol{\xi}\|^2} \tag{A.7}$$

Substituting (A.7) in (A.5) one obtains

$$x_m = v_m - \frac{\boldsymbol{\xi}^{\mathrm{T}}\boldsymbol{v} - W}{\|\boldsymbol{\xi}\|^2}\xi_m. \tag{A.8}$$

```
PositiveOrthantProjector $\Pi(\boldsymbol{v}, \boldsymbol{\xi}, W)$

$\boldsymbol{x} \leftarrow \boldsymbol{v}$                                    (P1)
$\overline{\boldsymbol{\xi}} \leftarrow \boldsymbol{\xi}$                                    (P2)
Repeat
  $\eta \leftarrow \frac{\overline{\boldsymbol{\xi}}^{\mathrm{T}}\boldsymbol{x} - W}{\overline{\boldsymbol{\xi}}^{\mathrm{T}}\overline{\boldsymbol{\xi}}}$                          (P3)
  $\boldsymbol{h} \leftarrow \boldsymbol{x} - \eta\overline{\boldsymbol{\xi}}$                           (P4)
  For $m \leftarrow 1$ To $M$
    $x_m \leftarrow \Theta(h_m) h_m$                   (P5)
    $\overline{\xi}_m \leftarrow \overline{\delta}(x_m)\xi_m$                   (P6)
  End
Until( $\forall m = 1, \ldots, M : x_m \geq 0$ )  (P7)
```

Figure A.11: Pseudocode for the optimal projection of an exterior point $\boldsymbol{v}$ onto the positive orthant fulfilling additionally the equality constraint $\boldsymbol{\xi}^{\mathrm{T}}\boldsymbol{x} = W$. After termination the final $\boldsymbol{x}$ represents the solution $\tilde{\boldsymbol{x}}$ to the optimization problem (A.1).

Considering arbitrary $\boldsymbol{v}$ vectors, the resulting x_m can be negative, thus violating the last line in (A.1). Therefore, the solution (A.8) must be successively changed to ensure $x_m \geq 0$. To this end, those x_m for which $x_m < 0$ holds are to be changed to zero, i.e., $x_m < 0 \Rightarrow x_m = 0$. While this change ensures the $x_m \geq 0$ condition, the equality constraint for the new $\boldsymbol{x}$ vector $\boldsymbol{\xi}^{\mathrm{T}}\boldsymbol{x} = W$ will be violated. Therefore, one has to repeat the optimization problem (A.1) for that subspace that does not contain those components of $\mathbf{x}$ that have been set to zero (the latter do not contribute to the $\boldsymbol{x}^{\mathrm{T}}\boldsymbol{\xi}$ scalar product). In order to formalize this iterative process that stops after at most M steps, let us introduce the anti-delta function $\overline{\delta}$

$$\overline{\delta}(x) := \begin{cases} 1, & \text{if } x \neq 0, \\ 0, & \text{if } x = 0 \end{cases} \tag{A.9}$$

and the step function Θ

$$\Theta(x) := \begin{cases} 1, & \text{if } x \geq 0, \\ 0, & \text{if } x < 0. \end{cases} \tag{A.10}$$

The algorithm is given in Fig. A.11. In the first Repeat-Until-cycle, projection is done according to (A.7) and (A.8) in Lines (P3) and (P4). However, this might have produced $\boldsymbol{h}$ components less than zero. Therefore, those components are set to zero in (P5). These components are no longer regarded in the optimization process. That is, the optimization is further performed in the complement subspace. This means that η and x_m update in (P3) and (P4) must take into account only the nonzero components. This can be accomplished by zeroing the respective components in $\boldsymbol{\xi}$ in Line (P6). This yields the new $\overline{\boldsymbol{\xi}}$ vector to be used instead of $\boldsymbol{\xi}$ in (A.7), (A.8) and (P3), (P4), respectively. This process of zeroing negative $\boldsymbol{h}$ components and projecting again in the remaining subspace is performed until there is no negative x_m left. The final $\boldsymbol{x}$ vector is then the constrained minimizer of (A.1).

References

[1] H.-G. Beyer, S. Finck, On the Design of Constraint Covariance Matrix Self-Adaptation Evolution Strategies Including a Cardinality Constraint, IEEE Transactions on Evolutionary Computation 16 (4) (2012) 578–596.

[2] J. Birge, F. Louveaux, Introduction to Stochastic Programming, Springer-Verlag, New York, 1997.

[3] G. C. Pflug, W. Römisch, Modeling, Measuring and Managing Risk, World Scientific, Singapore, 2007.

[4] G. C. Pflug, Optimal Scenario Tree Generation for Multiperiod Financial Planning, Mathematical Programming 89 (2001) 251–271.

[5] H. Heitsch, W. Römisch, Scenario Reduction Algorithms in Stochastic Programming, Computational Optimization and Applications 24 (2003) 187–206.

[6] P. Klaassen, Discretized Reality and Spurious Profits in Stochastic Programming Models for Asset/Liability Management, European Journal of Operational Research 101 (1997) 374–392.

[7] A. Geyer, M. Hanke, A. Weissensteiner, No-arbitrage Conditions, Scenario Tress, and Multi-asset Financial Optimization, European Journal of Operational Research 206 (2010) 609–613.

[8] M. Magill, G. Constantinides, Portfolio Selection with Transaction Costs, Journal of Economic Theory Theory 13 (1976) 245–263.

[9] A. Morton, S. Pliska, Optimal Portfolio Management with Fixed Transaction Costs, Mathematical Finance 5 (1995) 337–356.

[10] R. Korn, Optimal Portfolios: Stochastic Models for Optimal Investment and Risk Management in Continuous Time, World Scientific, Singapore, 1998.

[11] A. Cadenillas, Consumption-investment Problems with Transaction Costs: Survey and Open Problems, Mathematical Methods of Operations Research 51 (2000) 43–68.

[12] Y. Kabanov, C. Klüppelberg, A Geometric Approach to Portfolio Optimization in Models with Transaction Costs, Finance Stochastics 8 (2004) 207–227.

[13] K. Muthuraman, S. Kumar, Multidimensional Portfolio Optimization with Proportional Transaction Costs, Mathematical Finance 16 (2006) 301–335.

[14] B. C. on Banking Supervision, Amendment to the Capital Accord to Incorporate Market Risks, Bank for International Settlements (2005).

[15] P. Artzner, F. Delbaen, J.-M. Eber, D. Heath, Coherent Measures of Risk, Mathematical Finance 9 (1999) 203–228.

[16] H. Föllmer, A. Schied, Convex Measures of Risk and Trading Constraints, Finance and Stochastics 6 (2002) 429–447.

[17] R. T. Rockafellar, S. Uryasev, Optimization of Conditional Value-at-risk, Journal of Risk 2 (3) (2000) 21–41.

[18] P. Artzner, F. Delbaen, J. M. Eber, D. Heath, H. Ku, Coherent Multiperiod Risk Adjusted Values and Bellman's Principle, Annals of Operations Research 152 (2007) 5–22.

[19] P. Cheridito, F. Delbaen, M. Kupper, Coherent and Convex Risk Measures for Bounded càdlàg Processes, Stochastic Processes and Applications 112 (2004) 1–22.

[20] S. Weber, Distribution-invariant Dynamic Risk Measures, Information and Dynamic Consistency, Mathematical Finance 16 (2006) 419–441.

[21] H.-G. Beyer, B. Sendhoff, Covariance Matrix Adaptation Revisited – the CMSA Evolution Strategy, in: G. Rudolph et al. (Ed.), Parallel Problem Solving from Nature 10, Springer, Berlin, 2008, pp. 123–132.

[22] N. Hansen, S. Müller, P. Koumoutsakos, Reducing the Time Complexity of the Derandomized Evolution Strategy with Covariance Matrix Adaptation (CMA-ES), Evolutionary Computation 11 (1) (2003) 1–18.

[23] D. Arnold, Analysis of a Repair Mechanism for the $(1, \lambda)$-ES Applied to a simple constrained problem, in: J. Krasnogor et al. (Ed.), GECCO-2011: Proceedings of the Genetic and Evolutionary Computation Conference, ACM, New York, 2011, pp. 853–860.

[24] L. Systems, LINGO user's guide, Lindo Systems Inc., Chicago, 2011.

[25] C. Coello Coello, Theoretical and Numerical Constraint-handling Techniques used with Evolutionary Algorithms: a Survey of the State of the Art, Computer Methods in Applied Mechanics and Engineering 191 (11–12) (2002) 1245–1287.

[26] O. Kramer, A Review of Constraint-Handling Techniques for Evolution Strategies., Applied Computational Intelligence and Soft Computing (2010) 11 pages.

[27] T. Tometzki, S. Engell, Systematic Initialization Techniques for Hybrid Evolutionary Algorithms for Solving Two-Stage Stochastic Mixed-Integer Programs, IEEE Transactions on Evolutionary Computation 15 (2) (2011) 196–214.

[28] H.-G. Beyer, B. Sendhoff, Robust Optimization - A Comprehensive Survey, Computer Methods in Applied Mechanics and Engineering 196 (33–34) (2007) 3190–3218.

[29] J. Duchi, S. Shalev-Shwartz, Y. Singer, T. Chandra, Efficient Projections onto the ℓ_1-Ball for Learning in High Dimensions, in: Proceedings of the 25th International Conference on Machine Learning, ICML '08, ACM, New York, NY, USA, 2008, pp. 272–279.