Chapter 12
Applying Markowitz's Critical Line Algorithm

Andras Niedermayer and Daniel Niedermayer

12.1 Introduction

In recent years, Monte Carlo methods used for portfolio resampling have become increasingly popular in financial engineering and research.[1] When lower and upper bounds on portfolio holdings are imposed, these methods repeatedly employ quadratic optimization algorithms and, therefore, require a fast implementation of the mean variance optimizer. However, fast implementations are not publicly available. This is surprising given that with some simple numerical improvements an implementation of Markowitz's (1952) original Critical Line Algorithm (CLA) significantly outperforms standard software packages and a recently developed quadratic optimization algorithm described in Steuer et al. (2006). Therefore, the aim of this article is to derive all steps needed by the CLA and also to provide a didactic alternative to the framework in Markowitz and Todd (2000). The derivation of the equations and the pseudocode in the Appendix should ease the implementation of the fast version of the CLA by researchers and financial practitioners alike.

As a benchmark for computational speed, we use the results in Steuer et al. (2006). They develop a simplex-based algorithm that calculates all turning points of the constrained minimum variance frontier (CMVF) while significantly reducing computational time compared to standard software packages such as Matlab, Cplex, LINGO, Mathematica, and premium Solver. In order to compare their results with the CLA, we implement a numerically enhanced version of the algorithm in Fortran 90. We show that this algorithm outperforms the algorithm in Steuer et al. (2006) by a factor of almost 10,000 (for 2,000 assets) and standard software packages by even more.

A. Niedermayer (✉)
Kellogg School of Management, Managerial Economics and Decision Sciences, Northwestern University, Evanston, IL 60208, USA
e-mail: a-niedermayer@kellogg.northwestern.edu

D. Niedermayer
Credit Suisse, Asset Management and WWZ, University of Basel, Holbeinstrasse 12, CH-4051 Basel, Switzerland

[1] Resampling simulations have been introduced in, e.g., Jorion (1992) and Michaud (1998) and studied in, e.g., Scherer (2002), Markowitz and Usmen (2003), Scherer and Martin (2006), and Wolf (2006).

J.B. Guerard (ed.), *Handbook of Portfolio Construction: Contemporary Applications of Markowitz Techniques*, DOI 10.1007/978-0-387-77439-8_12,

From this observation, we conclude that the high performance of the CLA is not well known. In fact, excluding the paper by Steuer et al. (2006), no studies benchmarking quadratic optimization algorithms' performance are known to us. Moreover, as no publicly available software package exists that computes the entire CMVF, we provide a Matlab optimization package using our Fortran 90 implementation of the CLA.[2]

Finally, this paper can be considered as a didactic alternative to the standard CLA as presented by Markowitz. All numerical improvements to the algorithm are treated explicitly.

The rest of the paper is organized as follows: Sect. 12.2 introduces the mathematical framework and definitions required by the CLA. Section 12.3 formulates the quadratic optimization method and the numerical improvement. Section 12.4 describes performance tests and computational experience. Section 12.5 presents the conclusions. Proofs and a pseudocode representation are given in the Appendix.

12.2 The Framework

Given is a universe of n assets whose returns have the expected value $\boldsymbol{\mu}$ (n vector) and the $n \times n$ positive definite covariance matrix $\boldsymbol{\Sigma}$. In the following, we will denote an investor's weights assigned to each asset by the n vector $\boldsymbol{w}$ with the components summing up to 1.

For a minimum variance portfolio where lower and upper bounds on asset weights are included, we define an n vector containing the asset weights' lower bounds ($w_i \geq l_i, \quad \forall i$) $\boldsymbol{l}$; an n vector containing the asset weights' upper bounds ($w_i \leq u_i, \quad \forall i$) $\boldsymbol{u}$; and $\mathbb{F}$, a subset of $N = \{1, 2, \ldots, n\}$ containing all assets' indices where weights are within their bounds ($l_i < w_i < u_i$) and its size $k \equiv |\mathbb{F}|$. We shall call the corresponding assets as *free assets*. We further define $\mathbb{B}$, the subset of all asset indices where the weights lie on one of their bounds ($w_i = l_i$ or $w_i = u_i$). Thus, $\mathbb{B} = \{1, 2, \ldots, n\} \setminus \mathbb{F}$. The sets of assets on their upper and lower bounds will be called $\mathbb{U}$ and $\mathbb{L}$, respectively, with $\mathbb{B} = \mathbb{U} \cup \mathbb{L}$.

When writing the free assets' indices at the beginning, the covariance matrix $\boldsymbol{\Sigma}$, the expected return vector $\boldsymbol{\mu}$, and the weight vector $\boldsymbol{w}$ can be subdivided into

$$\boldsymbol{\Sigma} = \begin{bmatrix} \boldsymbol{\Sigma}_F & \boldsymbol{\Sigma}_{FB} \\ \boldsymbol{\Sigma}_{BF} & \boldsymbol{\Sigma}_B \end{bmatrix}, \qquad \boldsymbol{\mu} = \begin{bmatrix} \boldsymbol{\mu}_F \\ \boldsymbol{\mu}_B \end{bmatrix}, \quad \text{and} \quad \boldsymbol{w} = \begin{bmatrix} \boldsymbol{w}_F \\ \boldsymbol{w}_B \end{bmatrix} \tag{12.1}$$

with the covariance matrices $\boldsymbol{\Sigma}_F$ and $\boldsymbol{\Sigma}_B$ of dimensions $(k \times k)$ and $(n-k) \times (n-k)$, the matrices $\boldsymbol{\Sigma}_{FB}$ and $\boldsymbol{\Sigma}_{BF}$ of dimensions $k \times (n-k)$ and $(n-k) \times k$, two k vectors $\boldsymbol{\mu}_F$ and $\boldsymbol{w}_F$, and two $(n-k)$ vectors $\boldsymbol{\mu}_B$ and $\boldsymbol{w}_B$. Moreover, from the symmetry of $\boldsymbol{\Sigma}$ follows $\boldsymbol{\Sigma}_{BF} = \boldsymbol{\Sigma}'_{FB}$.

[2] We also provide an interface for the open source Numerical Python system on Linux.

12.2.1 Unconstrained Case

Before turning to the constrained variance minimization, it is worth to familiarize oneself again with the unconstrained portfolio minimization problem. The unconstrained problem can be solved by means of the Lagrange function

$$L = \frac{1}{2}\boldsymbol{w}'\boldsymbol{\Sigma}\boldsymbol{w} - \gamma(\boldsymbol{w}'\mathbf{1} - 1) - \lambda(\boldsymbol{w}'\boldsymbol{\mu} - \mu_{\mathrm{p}}), \tag{12.2}$$

with the Lagrange coefficients γ and λ and the target expected return level μ_{p}.[3] The first constraint in (12.2) ensures that assets' weights sum to one; the second constraint tells that the portfolio's expected return equals the target level μ_{p}. Differentiating with respect to $\boldsymbol{w}$, γ, and λ and setting the results to zero, one obtains a system of $(n + 2)$ linear equations. Solving this system leads to the solution of the variance minimizing weight vector $\boldsymbol{w}^*$. Obviously, $\boldsymbol{w}^*$ will not generally satisfy the constraints $l_i \leq w_i \leq u_i$.

12.2.2 Constrained Case

The computation of efficient portfolios becomes more difficult when inequality constraints on asset holdings are included. If short selling of assets is forbidden, asset weights must be nonnegative and the constraint has the form of $\boldsymbol{w} \geq 0$.

However, some problems require a more general constraint with upper and lower bounds. For such problems, the optimal solution will be a portfolio where assets' weights lie within their respective bounds, thus, $l_i \leq w_i \leq u_i$ for all i. In the following, we shall call the solution of this problem a constrained minimum variance portfolio[4] and the graphical representation of the set of solutions in the $(\mu_{\mathrm{p}}, \sigma_{\mathrm{p}})$ plane as CMVF.

One important feature of the CMVF is the existence of turning points.[5]

Definition 12.1. A constrained minimum variance portfolio is called *turning point* if other constrained minimum variance portfolios in its vicinity contain different free assets. □

When knowing which assets at a certain expected return level μ_{p} are free in the constrained minimum variance portfolio, thus, knowing $\mathbb{F}$, the problem can be formulated easily. This is stated in the following proposition.

[3] In the following, we will denote a vector of ones $(1, \ldots, 1)'$ as $\mathbf{1}$. To emphasize that the size of a vector $\mathbf{1}$ is the same as of a set $\mathbb{A}$, we will write $\mathbf{1}_{\boldsymbol{A}}$ (e.g., $\mathbf{1}_{\boldsymbol{F}}$ has size k and $\mathbf{1}_{\boldsymbol{B}}$ has size $n - k$).

[4] This is also called feasible mean variance efficient portfolio in the literature.

[5] In Markowitz (1959), turning points are described as the intersections of two critical lines.

Proposition 12.1. *The free weights in the solution $\boldsymbol{w}^*$ of the constrained case are equal to the weights in the solution of the unconstrained case with the Lagrange function*

$$\begin{aligned} L = & \frac{1}{2}\begin{bmatrix} \boldsymbol{w}_F \\ \boldsymbol{w}_B \end{bmatrix}' \begin{bmatrix} \boldsymbol{\Sigma}_F & \boldsymbol{\Sigma}_{FB} \\ \boldsymbol{\Sigma}_{BF} & \boldsymbol{\Sigma}_B \end{bmatrix} \begin{bmatrix} \boldsymbol{w}_F \\ \boldsymbol{w}_B \end{bmatrix} \\ & - \gamma \left(\begin{bmatrix} \boldsymbol{w}_F \\ \boldsymbol{w}_B \end{bmatrix}' \begin{bmatrix} \mathbf{1}_F \\ \mathbf{1}_B \end{bmatrix} - 1 \right) - \lambda \left(\begin{bmatrix} \boldsymbol{w}_F \\ \boldsymbol{w}_B \end{bmatrix}' \begin{bmatrix} \boldsymbol{\mu}_F \\ \boldsymbol{\mu}_B \end{bmatrix} - \mu_\mathrm{p} \right), \end{aligned} \tag{12.3}$$

where the components $\boldsymbol{w}_F$ are subject to minimization and the components $\boldsymbol{w}_B$ are fixed to their actual values.

Proof. This is obvious: changing $\boldsymbol{w}_F$ infinitesimally ensures that all weights remain free. This cannot lead to a smaller L otherwise $\boldsymbol{w}^*$ would not be a solution of the constrained case. □

There is a major difference between (12.2) and (12.3); the underlying subsets in (12.3) depend on the constrained minimum variance portfolio's location. Since the subsets $\mathbb{F}$ and $\mathbb{B}$ do not change between turning points, the solutions between two turning points will be the solution of an unconstrained optimization upon the subset $\mathbb{F}$.

Corollary 12.1. *Combining two neighboring turning points with a real weight $\omega \in [0, 1]$ always leads to a constrained minimum variance portfolio.*

Proof. This follows from Proposition 12.1 and the fact that a linear combination of two solutions (for different values of μ_p) of the unconstrained problem is a solution as well. □

Differentiating (12.3) with respect to $\boldsymbol{w}_F$ yields

$$\boldsymbol{\Sigma}_F \boldsymbol{w}_F + \boldsymbol{\Sigma}_{FB} \boldsymbol{w}_B - \lambda \boldsymbol{\mu}_F = \gamma \mathbf{1}_F \,. \tag{12.4}$$

At this point the constraint $\boldsymbol{w}'\mathbf{1} = 1$ must be adapted according to (12.1) leading to

$$\mathbf{1}_F' \boldsymbol{w}_F = 1 - \mathbf{1}_B' \boldsymbol{w}_B \,. \tag{12.5}$$

Solving (12.4) together with (12.5) for γ yields

$$\gamma = -\lambda \frac{\mathbf{1}_F' \boldsymbol{\Sigma}_F^{-1} \boldsymbol{\mu}_F}{\mathbf{1}_F' \boldsymbol{\Sigma}_F^{-1} \mathbf{1}_F} + \frac{1 - \mathbf{1}_B' \boldsymbol{w}_B + \mathbf{1}_F' \boldsymbol{\Sigma}_F^{-1} \boldsymbol{\Sigma}_{FB} \boldsymbol{w}_B}{\mathbf{1}_F' \boldsymbol{\Sigma}_F^{-1} \mathbf{1}_F} \,. \tag{12.6}$$

Note that here λ is set exogenously instead of μ_p. Therefore, λ determines the value of γ and finally the expected return of the minimum variance portfolio. The value of μ_p is fictitious in (12.3) and the optimal solution is solely determined by λ. In fact, it is very similar to set λ exogenously or to calculate with a fixed μ_p and look at λ as Lagrange multiplier. This is because λ and $\boldsymbol{\mu}'\boldsymbol{w}$ $(= \mu_\mathrm{p})$ are linearly related between

turning points and because a higher λ yields a (constrained) minimum variance portfolio with higher expected return. This is stated here by Proposition 12.2 with the proof being banned to the Appendix.

Proposition 12.2. *Between two turning points, λ and $\boldsymbol{\mu}'\boldsymbol{w}$ are linearly related with a positive slope*

$$\frac{\partial(\boldsymbol{\mu}'\boldsymbol{w}(\lambda))}{\partial\lambda} > 0\,. \qquad \square$$

12.3 The Algorithm

The main idea for the algorithm presented is the following: first, the turning point with the highest expected return value is found; then the next lower turning point is calculated. This is illustrated in Fig. 12.1.

From the definition of a turning point, we know that each of them will differ in the composition of its free assets. Therefore, for each of them (12.4) will hold for a different subset $\mathbb{F}$. Except for the case where two or more turning points lie upon each other,[6] when passing a turning point one has to add or remove exactly one element from the set of free assets $\mathbb{F}$.

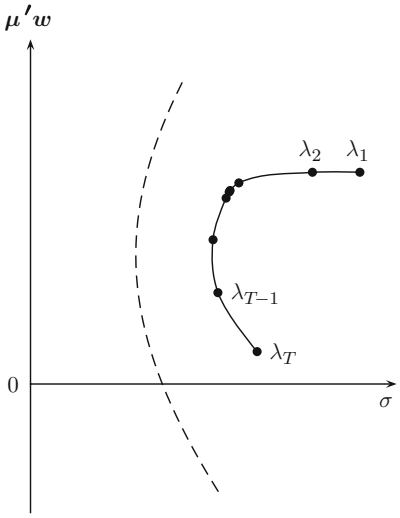

Fig. 12.1 This figure shows the minimum variance frontier (*dashed line*) and the constrained minimum variance frontier for ten assets and an arbitrarily chosen nonsingular covariance matrix. The *dots* represent the constrained frontier's turning points

[6] We do not discuss this possibility even though the algorithm can cope with it; when calculating numerically, there will hardly be two or more turning points on one (σ, μ) location. Markowitz (1959) proposes as a solution to this situation to either alter the μ of one asset slightly or to use the method described in Markowitz (1956).

Moreover, when moving downward from a turning point to the next one, λ will decrease (see Proposition 12.2). When looking at turning points such as in Fig. 12.1 it must, therefore, be that

$$\lambda_1 > \lambda_2 > \lambda_3 > \cdots > \lambda_T$$

with T being the number of turning points.

Sections 12.3.1 and 12.3.2 show how to find the turning point with the highest expected return (turning point 1) and how to move to the next lower turning point. Section 12.3.3 shows a way how to improve the algorithm's performance significantly.

12.3.1 The Starting Solution

This algorithm requires an initial solution on the CMVF. It is convenient to find the turning point with the highest expected return before moving to the next lower turning point.

Therefore, we order all assets with respect to their expected return values such that the first asset has the highest and the last asset the lowest expected return. After setting all asset weights to their lower bounds ($w_i = l_i$), we start to increase the weight of the first asset. If the upper bound is reached and $\boldsymbol{w}'\mathbf{1} < 1$, we start to increase the second assets' weight, and so forth. This procedure terminates when $\boldsymbol{w}'\mathbf{1} = 1$ is reached.[7]

The solution is typically a weight vector where the weights of the first assets are set to their upper bounds and the last assets' weights are set to their lower bounds. There is one asset in the middle, where the weight is between its bounds. We will call this asset the free asset and index it with i_{free}. The weight of the free asset is $1 - \sum_{i \in \mathbb{U}} w_i - \sum_{i \in \mathbb{L}} w_i \equiv 1 - \boldsymbol{w}_{\boldsymbol{B}}'\mathbf{1}_{\boldsymbol{B}}$.

We solve a simpler problem than Markowitz and Todd (2000) as we have the only equality constraint $\boldsymbol{w}'\mathbf{1} = 1$, whereas Markowitz and Todd consider the general constraint $\boldsymbol{A}\boldsymbol{w} = \boldsymbol{b}$. However, because in many practical situations the specific case $\boldsymbol{w}'\mathbf{1} = 1$ is sufficient, one can use our simpler algorithm to find the first turning point instead of the simplex algorithm used in Markowitz and Todd (2000).

12.3.2 Iteration

When moving from a turning point to the next lower one by decreasing λ, one of the following two situations will occur; either one free asset moves to one of its

[7] Obviously, this requires $\mathbf{1}'\boldsymbol{l} \leq 1 \leqslant \mathbf{1}'\boldsymbol{u}$. For $\mathbf{1}'\boldsymbol{u} < 1$ or $\mathbf{1}'\boldsymbol{l} > 1$, there is no solution to the portfolio optimization problem. For $\mathbf{1}'\boldsymbol{u} = 1$ or $\mathbf{1}'\boldsymbol{l} = 1$, the whole efficient frontier consists of only one portfolio, namely $\boldsymbol{w} = \boldsymbol{u}$ or $\boldsymbol{w} = \boldsymbol{l}$, respectively.

bounds or an asset formerly on its bound becomes free. These two cases have to be considered in order to compute the next turning point's λ and $\boldsymbol{w}$.

12.3.2.1 Case (a) One Formerly Free Asset Moves to Its Bound

Let λ_{current} belong to a turning point and let $\mathbb{F}$ be the set of the free assets slightly below this turning point (i.e., for λ such that $\lambda_{\text{current}} \equiv \lambda_t > \lambda > \lambda_{t+1}$).

For this subset containing k variables (12.4) holds and thus

$$\boldsymbol{w}_F = -\boldsymbol{\Sigma}_F^{-1}\boldsymbol{\Sigma}_{FB}\boldsymbol{w}_B + \gamma\boldsymbol{\Sigma}_F^{-1}\mathbf{1}_F + \lambda\boldsymbol{\Sigma}_F^{-1}\boldsymbol{\mu}_F\,. \tag{12.7}$$

Substituting γ from (12.6) into (12.7) gives us $\boldsymbol{w}_F$ as a linear function of λ. When decreasing λ, the asset i will hit the lower bound if the derivative $\mathrm{d}w_i/\mathrm{d}\lambda$ is positive and hit the upper bound if the derivative is negative. We denote the value of λ as $\lambda^{(i)}$ at the point where asset i hits the corresponding bound. $\lambda^{(i)}$ can be derived from the linear relation between $\boldsymbol{w}_{Fi}$ and λ resulting from (12.6) and (12.7). This gives

$$\begin{aligned}\lambda^{(i)} = &\frac{1}{C_i}\left[\left(1 - \mathbf{1}_B'\boldsymbol{w}_B + \mathbf{1}_F'\boldsymbol{\Sigma}_F^{-1}\boldsymbol{\Sigma}_{FB}\boldsymbol{w}_B\right)(\boldsymbol{\Sigma}_F^{-1}\mathbf{1}_F)_i\right.\\ &\left.-(\mathbf{1}_F'\boldsymbol{\Sigma}_F^{-1}\mathbf{1}_F)\left(b_i + (\boldsymbol{\Sigma}_F^{-1}\boldsymbol{\Sigma}_{FB}\boldsymbol{w}_B)_i\right)\right]\end{aligned} \tag{12.8}$$

with

$$C_i = -(\mathbf{1}_F'\boldsymbol{\Sigma}_F^{-1}\mathbf{1}_F)(\boldsymbol{\Sigma}_F^{-1}\boldsymbol{\mu}_F)_i + (\mathbf{1}_F'\boldsymbol{\Sigma}_F^{-1}\boldsymbol{\mu}_F)(\boldsymbol{\Sigma}_F^{-1}\mathbf{1}_F)_i \tag{12.9}$$

and[8]

$$b_i = \begin{cases} u_i & \text{if } C_i > 0\,,\\ l_i & \text{if } C_i < 0\,. \end{cases} \tag{12.10}$$

Note that for $k = 1$ one gets $C_i = 0$, which reflects the fact that the constraint $\mathbf{1}_F'\boldsymbol{w}_F = w_i = 1 - \mathbf{1}_B'\boldsymbol{w}_B$ uniquely determines w_i. Therefore, case (a) should be considered only for $k > 1$. C_i is zero for all i if accidentally $\boldsymbol{\mu}_F$ is proportional to $\mathbf{1}_F$, i.e., $\mu_i = \mu_j$ for all $i, j \in \mathbb{F}$.

The next $\lambda < \lambda_{\text{current}}$ where an asset wants to leave the subset $\mathbb{F}$ is

$$\lambda_{\text{inside}} = \max_{i\in\mathbb{F}}\{\lambda^{(i)}\}\,, \tag{12.11}$$

or λ_{inside} does not exist if $k = 1$ or $C_i = 0$ for all i.

[8] Note that $\mathrm{d}w_i/\mathrm{d}\lambda = -C_i/(\mathbf{1}_F'\boldsymbol{\Sigma}_F^{-1}\mathbf{1}_F)$. Also note further that $i \in \mathbb{F} = \{i_1, i_2, \ldots, i_k\}$ and, therefore, i can take values from 1 to n (and not from 1 to k).

However, λ_{inside} will only describe the next lower turning point if there is no portfolio with a λ where $\lambda_{\text{current}} > \lambda > \lambda_{\text{inside}}$ and where an asset on its bound wants to get into the subset $\mathbb{F}$. This situation is summarized by case (b).

12.3.2.2 Case (b) One Asset Formerly on Its Bound Wants to Become Free

When moving downward in $\boldsymbol{\mu}'\boldsymbol{w}$ it might occur that an asset i formerly on its bound wants to become free. The corresponding portfolio is, therefore, a turning point where the subsets $\mathbb{F}$ and $\mathbb{B}$ have to be redefined. Let us denote the new subsets as

$$\begin{aligned}\mathbb{F}_i &\equiv \mathbb{F} \cup \{i\},\\ \mathbb{B}_i &\equiv \mathbb{B} \setminus \{i\},\end{aligned}$$

where $i \in \mathbb{B}$. Analogously to (12.8) the value $\lambda^{(i)}$ where the newly included asset i's weight moves away from its bound is given by

$$\begin{aligned}\lambda^{(i)} = \frac{1}{C_i}\Big[&\Big(1 - \mathbf{1}'_{\boldsymbol{B}_i}\boldsymbol{w}_{\boldsymbol{B}_i} + \mathbf{1}'_{\boldsymbol{F}_i}\boldsymbol{\Sigma}_{\boldsymbol{F}_i}^{-1}\boldsymbol{\Sigma}_{\boldsymbol{F}_i\boldsymbol{B}_i}\boldsymbol{w}_{\boldsymbol{B}_i}\Big)(\boldsymbol{\Sigma}_{\boldsymbol{F}_i}^{-1}\mathbf{1}_{\boldsymbol{F}_i})_i\\ &-(\mathbf{1}'_{\boldsymbol{F}_i}\boldsymbol{\Sigma}_{\boldsymbol{F}_i}{}^{-1}\mathbf{1}_{\boldsymbol{F}_i})\Big(b_i + (\boldsymbol{\Sigma}_{\boldsymbol{F}_i}^{-1}\boldsymbol{\Sigma}_{\boldsymbol{F}_i\boldsymbol{B}_i}\boldsymbol{w}_{\boldsymbol{B}_i})_i\Big)\Big]\end{aligned} \tag{12.12}$$

with

$$C_i = -(\mathbf{1}'_{\boldsymbol{F}_i}\boldsymbol{\Sigma}_{\boldsymbol{F}_i}^{-1}\mathbf{1}_{\boldsymbol{F}_i})(\boldsymbol{\Sigma}_{\boldsymbol{F}_i}^{-1}\boldsymbol{\mu}_{\boldsymbol{F}_i})_i + (\mathbf{1}'_{\boldsymbol{F}_i}\boldsymbol{\Sigma}_{\boldsymbol{F}_i}^{-1}\boldsymbol{\mu}_{\boldsymbol{F}_i})(\boldsymbol{\Sigma}_{\boldsymbol{F}_i}^{-1}\mathbf{1}_{\boldsymbol{F}_i})_i \tag{12.13}$$

and where for convenience b_i is used for $(\boldsymbol{w}_{\boldsymbol{F}_i})_i = w_i$, which is an asset on its bound possibly becoming free. If asset i was previously on its upper bound, b_i stands for u_i and if it was on the lower bound, it stands for l_i.[9]

In order to find the maximal $\lambda^{(i)} < \lambda_{\text{current}}$ where an asset i currently on its bound wants to become free, (12.12) must be applied for all $i \in \mathbb{B}$.

$$\lambda_{\text{outside}} = \max_{i\in\mathbb{B}}\{\lambda^{(i)} \mid \lambda^{(i)} < \lambda_{\text{current}}\}. \tag{12.14}$$

Again, if no $\lambda^{(i)} < \lambda_{\text{current}}$ exists, we remember that there is no solution for λ_{outside}.

[9] To avoid identifying the trivial solution $\lambda^{(i)} = \lambda_{\text{current}}$ erroneously as a turning point if asset i just went out in the previous step one can check the derivative $\mathrm{d}w_i/\mathrm{d}\lambda$ similarly to case (a). If the derivative is negative and the asset was previously on the upper bound, we have the λ_{current} of the previous turning point and numerical imprecision has led to $\lambda^{(i)} = \lambda_{\text{current}} - \epsilon$ with ϵ small but positive.

12.3.2.3 Finding the Next Turning Point

In order to find out which case will occur, the values of λ_{inside} and λ_{outside} must be compared.

- If solutions for both λ_{inside} and λ_{outside} could be found, then the next turning point will have a λ defined as

$$\lambda_{\text{new}} = \max\{\lambda_{\text{inside}}, \lambda_{\text{outside}}\}.$$

  Thus, e.g., case (a) is characterized by $\lambda_{\text{inside}} > \lambda_{\text{outside}}$.
- If a solution only for λ_{inside} or λ_{outside} could be found, λ_{new} is overwritten by the respective value.
- Depending on which case occurs, we replace $\mathbb{F}$ by $\mathbb{F} \setminus \{i\}$ or by $\mathbb{F}_i$, $\mathbb{B}$ by $\mathbb{B} \cup \{i\}$ or $\mathbb{B}_i$, and λ_{current} by λ_{new}.
- If no solution for λ_{inside} and λ_{outside} could be found, we have reached the lowest turning point and the algorithm terminates.[10]

The pseudocode in Appendix summarizes the algorithm for the simplified case when $l_i = 0$ and $u_i = +\infty$.[11]

One way of checking the results is to look at the last turning point's weight vector. This weight vector must be the "opposite" one to the initial solution. When ordering all weights with regard to their expected returns, the last weights must be at their upper and the first weights at their lower bounds. The remaining weight will correspond to the free asset.

Note that in contrast to the calculations in (12.8), the specification of $\mathbb{F}_i$ in (12.12) depends on i and $\boldsymbol{\Sigma}_{\boldsymbol{F}_i}^{-\mathbf{1}}$ must be recalculated for each $i \notin \mathbb{F}$.[12] Moreover, as the next turning point has one asset more or one asset less in $\mathbb{F}$, the inverse of the respective covariance matrix $\boldsymbol{\Sigma}_{\boldsymbol{F}}^{-\mathbf{1}}$ must be recalculated each time. We will show in the following, how these time consuming computations of inverses can be avoided.

12.3.3 Improving Performance

In the algorithm described above, the compositions of $\mathbb{F}$ and $\mathbb{B}$ change with one asset being included or excluded. Here we show that one can avoid recalculating the inverse of the corresponding matrices each time.

---

[10] For practical purposes, the lower half of the efficient frontier (with μ decreasing and σ increasing) does not interest us. In this case, we can terminate the algorithm when σ starts increasing again which corresponds to $\lambda = 0$.

[11] Note that in this case $\boldsymbol{w}_{\boldsymbol{B}} = 0$ and the equations become much simpler. The starting solution is also simpler: one has to set the weight of the asset with the highest expected return to 1.

[12] Or at least the vectors $\boldsymbol{\Sigma}_{\boldsymbol{F}_i}^{-\mathbf{1}}\mathbf{1}_{\boldsymbol{F}_i}$ and $\boldsymbol{\Sigma}_{\boldsymbol{F}_i}^{-\mathbf{1}}\boldsymbol{\mu}_{\boldsymbol{F}_i}$ have to be calculated.

12.3.3.1 Expansion of the Covariance Matrix $\boldsymbol{\Sigma}_F$

Lemma 12.1. *Let $\boldsymbol{A}$ be a symmetric nonsingular $k \times k$ matrix, $\boldsymbol{a}$ a $k \times 1$ vector, and α be a scalar. Then for the expanded matrix's inverse*

$$\begin{bmatrix} \boldsymbol{A} & \boldsymbol{a} \\ \boldsymbol{a}' & \alpha \end{bmatrix}^{-1} = \begin{bmatrix} \boldsymbol{A}^{-1} + \beta \boldsymbol{c}\boldsymbol{c}' & -\beta \boldsymbol{c} \\ -\beta \boldsymbol{c}' & \beta \end{bmatrix} \tag{12.15}$$

holds where

$$\boldsymbol{c} = \boldsymbol{A}^{-1}\boldsymbol{a} \qquad \text{and} \qquad \beta = \frac{1}{\alpha - \boldsymbol{c}'\boldsymbol{a}}.$$

Proof. Multiplying the expanded matrix with the right-hand side of (12.15) *yields the identity matrix.* □

Our algorithm requires often expanding the subset $\mathbb{F}$ by one element i and recalculating the inverse of the covariance matrix, $\boldsymbol{\Sigma}_{F_i}^{-1}$, for the new subset. Lemma 12.1 frees us from the burden of making this calculation all over again. This reduces the number of operations for inverting $\begin{bmatrix} \boldsymbol{A} & \boldsymbol{a} \\ \boldsymbol{a}' & \alpha \end{bmatrix}$ from $k^3/3$ to $2k^2$.

12.3.3.2 Reduction of the Covariance Matrix $\boldsymbol{\Sigma}_F$

Reducing the covariance matrix by one row and column does not require the inversion of the newly obtained matrix either. Having calculated the inverse of the expanded covariance matrix as in the previous section and now deleting the given row and column, the newly obtained matrix's inverse can be calculated. This is stated in Lemma 12.2 where for presentational purposes the given index is assumed to be the last one.

Lemma 12.2. *Let $\boldsymbol{A}$ and $\boldsymbol{B}$ be $k \times k$ matrices, $\boldsymbol{a}$ and $\boldsymbol{b}$ k vectors, and α and β be two scalars. Then if*

$$\begin{bmatrix} \boldsymbol{A} & \boldsymbol{a} \\ \boldsymbol{a}' & \alpha \end{bmatrix}^{-1} = \begin{bmatrix} \boldsymbol{B} & \boldsymbol{b} \\ \boldsymbol{b}' & \beta \end{bmatrix} \tag{12.16}$$

holds then

$$\boldsymbol{A}^{-1} = \boldsymbol{B} - \frac{1}{\beta}\boldsymbol{b}\boldsymbol{b}'.$$

holds as well.

Proof. Combine (12.15) *and* (12.16) *and solve for $\boldsymbol{A}^{-1}$.* □

12.3.3.3 Key Improvement

The most remarkable improvement stems from the fact that in (12.12) we need to know neither $\boldsymbol{\Sigma}_{F_i}^{-1}$ nor $\boldsymbol{\Sigma}_{F_i}^{-1}\boldsymbol{\Sigma}_{F_i B_i}$. This is stated in Proposition 12.3.

Proposition 12.3. *Expression* (12.12)*,*

$$\lambda^{(i)} = \frac{1}{C_i}\left[\left(1 - \mathbf{1}'_{B_i} w_{B_i} + \mathbf{1}'_{F_i} \Sigma_{F_i}^{-1} \Sigma_{F_i B_i} w_{B_i}\right) (\Sigma_{F_i}^{-1} \mathbf{1}_{F_i})_i \right.$$
$$\left. -(\mathbf{1}'_{F_i} \Sigma_{F_i}^{-1} \mathbf{1}_{F_i}) \left(b_i + (\Sigma_{F_i}^{-1} \Sigma_{F_i B_i} w_{B_i})_i\right)\right]$$

with

$$C_i = -(\mathbf{1}'_{F_i} \Sigma_{F_i}^{-1} \mathbf{1}_{F_i})(\Sigma_{F_i}^{-1} \mu_{F_i})_i + (\mathbf{1}'_{F_i} \Sigma_{F_i}^{-1} \mu_{F_i})(\Sigma_{F_i}^{-1} \mathbf{1}_{F_i})_i$$

can be rewritten as

$$\lambda^{(i)} = \frac{1}{D_i}\left[\left(1 - \mathbf{1}'_B w_B + \mathbf{1}'_F \Sigma_F^{-1} \Sigma_{FB} w_B\right)(1 - a' \Sigma_F^{-1} \mathbf{1}_F)\right. \tag{12.17}$$
$$\left. + \left(a' \Sigma_F^{-1} \Sigma_{FB} w_B - \Sigma_{iB} w_B\right)(\mathbf{1}'_F \Sigma_F^{-1} \mathbf{1}_F)\right]$$

where the used variables are defined as

$$\Sigma_{F_i} = \begin{bmatrix} \Sigma_F & a \\ a' & \alpha \end{bmatrix}, \qquad \mu_{F_i} = \begin{bmatrix} \mu_F \\ \mu_i \end{bmatrix}$$

and

$$D_i = (1 - a' \Sigma_F^{-1} \mathbf{1}_F)(\mathbf{1}'_F \Sigma_F^{-1} \mu_F) - (\mu_i - a' \Sigma_F^{-1} \mu_F)(\mathbf{1}'_F \Sigma_F^{-1} \mathbf{1}_F).$$

Proof. The derivation is shown in the Appendix. □

It is remarkable that in (12.17) the vector a (the ith column of Σ corresponding to a trial $i \in \mathbb{B}$) enters linearly. Therefore, the calculation of $\lambda^{(i)}$ is fast and one should calculate only two scalar products with a for each $i \in \mathbb{B}$.

12.4 Performance Tests

Obviously, it is problematic to compare different algorithms based on absolute CPU times. Their performance will strongly depend on the programming language and on the algorithm's memory requirements. When not testing all algorithms on the same computer, different processor performance, RAM size, and system configurations do not allow for a simple interpretation of the differences in CPU run times.

Therefore, the following has to be kept in mind when comparing run times. First, when looking at the increase of CPU time at an increasing number of assets, the algorithms' relative performance is independent from the programming language and other hardware and software properties (as long as there are no memory bottlenecks). Second, the difference in the programming languages does not explain that amount of CPU time improvement such as obtained by our tests.

In the following, a Fortran 90 implementation (Fortran 90 CLA) of the discussed algorithm is tested against three programs for the case where the lower bound is zero and the upper bound is infinity; a simplex-like algorithm based on Wolfe (1959) coded in Java [Java Wolfe-Simplex as described and implemented in Niedermayer (2005)]; and the quadratic optimization package of Matlab. Furthermore, we compare our results with those in Steuer et al. (2006),[13] whose simplex-based multiparametric optimization algorithm was implemented in Java (Java MPQ). The latter comparison is important; as argued in Steuer et al. (2006), the MPQ outperforms Matlab, Cplex, LINGO, Mathematica, and Excel's premium Solver. Steuer et al. (2006) did not compare the Java MPQ algorithm to the Excel Optimizer by Markowitz and Todd (2000) due to the 256 column limitation of Excel. Finally, we also provide run times of the Excel Optimizer by Markowitz and Todd (2000). Note that this implementation is provided by Markowitz and Todd (2000) for illustrative purposes in form of an Excel VBA macro and can calculate the efficient frontier for up to 256 securities (the maximal number of columns in Excel). Note further that even though we ran the Optimizer with the same set of constraints as the other problems, it can solve the optimization problem for a more general set of constraints.

For the tests illustrated in Fig. 12.2, a positive definite covariance matrix was generated as

$$\boldsymbol{\Sigma} = \sum_{i=1}^{n} \boldsymbol{r}^{(i)} \boldsymbol{r}^{(i)\prime},$$

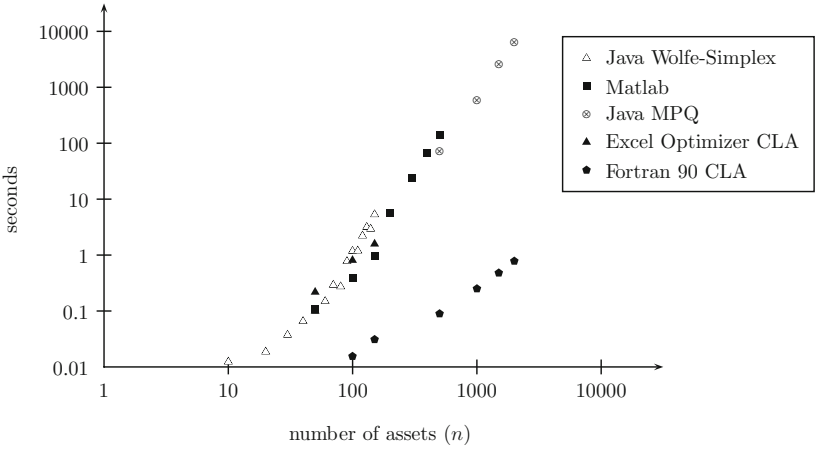

Fig. 12.2 Testing different algorithms for the case with lower bounds zero and upper bounds infinity: CPU times for different number of assets and randomly generated positive definite covariance matrix

[13] Similarly to Steuer et al. (2006), we ran our tests on a Dell 3.06-GHz desktop.

Table 12.1 Different CPU times in seconds for the case with lower bounds zero and upper bounds infinity

n	Fortran 90 CLA	Java Wolfe-Simplex	Matlab	Java MPQ	Excel optimizer CLA
50	–	0.10	0.109	–	0.219
100	0.0156	1.18	0.391	–	0.813
150	0.0312	5.35	0.985	–	1.578
500	0.09	–	141.6	72	–
1,000	0.25	–	–	602	–
1,500	0.48	–	–	2,580	–
2,000	0.78	–	–	6,300	–
Perf.	$\mathcal{O}(n^{1.6})$	$\mathcal{O}(n^{3.6})$	$\mathcal{O}(n^{3.2})$	$\mathcal{O}(n^{3.3})$	$\mathcal{O}(n^{1.8})$

The last row shows the estimates of the algorithms' performance with respect to the number of securities n. Note that the results of the MPQ performance stem from Hirschberger et al. (2004). Note further that the performance we have provided for the Fortran 90 CLA is calculated from the run times without matrix sizes 100 and 150 because for smaller matrix sizes the fixed costs of calculation seem to distort the data. When including matrix sizes 100 and 150 we get $\mathcal{O}(n^{1.3})$

where $\boldsymbol{r}^{(i)}$ is an n vector containing random numbers between [0, 1] and is regenerated for each i. Since our results and the MPQ results in Steuer et al. (2006) strongly depend on the number of free assets (i.e., assets not on their bounds), thus, of the maximum dimension, $\hat{k}$, of $\boldsymbol{\Sigma}_{\boldsymbol{F}}$ in (12.8) and (12.12), we made sure that $\hat{k}/n$ is similar to that in Steuer et al. (2006).[14] In our tests with 1,000 assets $\hat{k}$ was 60 and when testing with 2,000 assets $\hat{k}$ was 250.

In Fig. 12.2, both axes are logarithmic. The slope of the linear OLS fit corresponds, therefore, to the exponent of the respective algorithm's CPU time increase at an increasing number of assets. Note that the problem with Java Wolfe-Simplex is that the program's RAM requirements increase rapidly which allows only for the computation of problems up to 150 assets. The test's results are summarized in Table 12.1.

Since the Wolfe-Simplex algorithm and the Matlab quadratic optimization package only calculate one single point on the CMVF and do not calculate the whole frontier analytically such as our Fortran CLA algorithm, the Optimizer by Markowitz and Todd (2000), and the algorithm of Steuer et al. (2006), the CPU times reported in Table 12.1 support our method even more.

12.5 Conclusions

This paper presents the CLA developed by Markowitz and demonstrates its strong computational performance compared to standard software packages and to a recently published Optimization algorithm. We find that our implementation of the

[14] Generating $\boldsymbol{r}^{(i)} \sim u[-1/2, 1/2]$ would lead to different values of $\hat{k}/n$.

CLA[15] outperforms the current Matlab optimization tool by a factor of approximately 15,000 when the problem size (number of assets) is 2,000. When comparing with the algorithm in Steuer et al. (2006) that also computes all turning points analytically such as the CLA does, the performance improvement is still around 8,000.

In this paper, we treat all steps of the algorithm explicitly. The algorithm can be used for problems of large-scale portfolio optimization and CPU time-intensive Monte Carlo simulations. The pseudocode we provide is helpful for the implementation of the algorithm in other programming languages.

Appendix

Proofs

Proof (Proposition 12.2). For tractability we define three constants

$$C_{11} \equiv \mathbf{1}_F' \boldsymbol{\Sigma}_F^{-1} \mathbf{1}_F \,, \qquad C_{1\mu} \equiv \mathbf{1}_F' \boldsymbol{\Sigma}_F^{-1} \boldsymbol{\mu}_F \,, \qquad C_{\mu\mu} \equiv \boldsymbol{\mu}_F' \boldsymbol{\Sigma}_F^{-1} \boldsymbol{\mu}_F \,.$$

From (12.4) and the definitions given in (12.1) follows that

$$\boldsymbol{\mu}' \boldsymbol{w} = \boldsymbol{\mu}_F' \boldsymbol{w}_F + \boldsymbol{\mu}_B' \boldsymbol{w}_B = -\boldsymbol{\mu}_F' \boldsymbol{\Sigma}_F^{-1} \boldsymbol{\Sigma}_{FB} \boldsymbol{w}_B + \gamma C_{1\mu} + \lambda C_{\mu\mu} \,.$$

Differentiating this expression with respect to λ yields

$$\frac{\partial(\boldsymbol{\mu}' \boldsymbol{w})}{\partial \lambda} = C_{\mu\mu} - \frac{C_{1\mu}^2}{C_{11}} \,. \tag{12.18}$$

Since between two turning points $\boldsymbol{\Sigma}_F$ does not change, $\mu_{\mathrm{p}}(\lambda) = \boldsymbol{\mu}' \boldsymbol{w}(\lambda)$ is linear in λ with a slope given by (12.18). We show below that this slope is positive.

From the positive definiteness of $\boldsymbol{\Sigma}$ follows that its submatrix $\boldsymbol{\Sigma}_F$ and $\boldsymbol{\Sigma}_F^{-1}$ ($\equiv (\boldsymbol{\Sigma}_F)^{-1}$) are positive definite as well.

We introduce a vector $\boldsymbol{x} \equiv \mathbf{1}_F - \alpha \boldsymbol{\mu}_F$ with $\alpha \in \mathbb{R}$. Then $\boldsymbol{x}' \boldsymbol{\Sigma}_F^{-1} \boldsymbol{x}$ can be written as

$$(\mathbf{1}_F - \alpha \boldsymbol{\mu}_F)' \boldsymbol{\Sigma}_F^{-1} (\mathbf{1}_F - \alpha \boldsymbol{\mu}_F) = C_{11} - 2\alpha C_{1\mu} + \alpha^2 C_{\mu\mu} \,.$$

[15] The program is available on request from the authors in form of a Matlab package at http://www.niedermayer.ch/cla

Positive definiteness of $\mathbf{\Sigma}_F^{-1}$ means $\boldsymbol{x}'\mathbf{\Sigma}_F^{-1}\boldsymbol{x} > 0$ for any vector $\boldsymbol{x}$, hence, the equation $C_{11} - 2\alpha C_{1\mu} + \alpha^2 C_{\mu\mu} = 0$ cannot have a solution for α (unless $\boldsymbol{\mu}_F$ is parallel to $\mathbf{1}_F$). Therefore, the discriminant is negative which gives

$$C_{11}C_{\mu\mu} - C_{1\mu}^2 > 0\,.$$

□

Proof (Proposition 12.3). According to Lemma 12.1, $\mathbf{\Sigma}_{F_i}^{-1}$ can be expressed in terms of $\mathbf{\Sigma}_F^{-1}$, $\boldsymbol{a}$, and α. Multiplying $\mathbf{\Sigma}_{F_i}^{-1}$ by $\mathbf{1}_{F_i}$ and $\boldsymbol{\mu}_{F_i}$, respectively, yields

$$\mathbf{\Sigma}_{F_i}^{-1}\mathbf{1}_{F_i} = \begin{bmatrix} \mathbf{\Sigma}_F^{-1}\mathbf{1}_F - \beta(1 - \boldsymbol{c}'\mathbf{1})\boldsymbol{c} \\ \beta(1 - \boldsymbol{c}'\mathbf{1}) \end{bmatrix} \tag{12.19}$$

and

$$\mathbf{\Sigma}_{F_i}^{-1}\boldsymbol{\mu}_{F_i} = \begin{bmatrix} \mathbf{\Sigma}_F^{-1}\boldsymbol{\mu}_F - \beta(\mu_i - \boldsymbol{c}'\boldsymbol{\mu}_F)\boldsymbol{c} \\ \beta(\mu_i - \boldsymbol{c}'\boldsymbol{\mu}_F) \end{bmatrix}. \tag{12.20}$$

Multiplying (12.19) and (12.20) by $\mathbf{1}'_{F_i}$ and plugging the values into (12.12) yields the denominator in (12.17).

Using the following properties and Lemma 12.1 yields the numerator in (12.17):

$$\begin{aligned} \mathbf{1}'_{B_i}\boldsymbol{w}_{B_i} &= \mathbf{1}'_B\boldsymbol{w}_B - b_i\,, \\ \mathbf{\Sigma}_{F_iB_i}\boldsymbol{w}_{B_i} &= \begin{bmatrix} \mathbf{\Sigma}_{FB}\boldsymbol{w}_B - \boldsymbol{a}b_i \\ \mathbf{\Sigma}_{iB}\boldsymbol{w}_B - \alpha b_i \end{bmatrix}. \end{aligned}$$

□

Pseudocode

The following pseudocode describes the procedure which calculates all turning points of the minimum variance frontier for an arbitrary lower bound $\boldsymbol{l}$ and upper bound $\boldsymbol{u}$. Parameters are the expected returns $\boldsymbol{\mu}$ and the covariance matrix $\mathbf{\Sigma}$. Asset weights $\boldsymbol{w}^{(t)}$ are returned for each turning point t. Between turning points, weights are linear combinations of the two surrounding turning points.

In our notation, $x \leftarrow y$ means that the value y is assigned to the variable x. We define $\arg\max_i\{x_i\}$ to return i^* with $x_{i^*} \geq x_i$ for all i or NIL if the set $\{x_i\}$ is empty. $\max\{\cdot\}$ returns the greatest value unequal to NIL. As in the main text, $\mathbf{\Sigma}_F$ is a short-hand for the matrix $\{\Sigma_{ij}|i, j \in \mathbb{F}\}$ and $\boldsymbol{\mu}_F \equiv \{\mu_i|i \in \mathbb{F}\}$. The same applies to $\mathbf{\Sigma}_{F_i}$ and $\boldsymbol{\mu}_{F_i}$ with $\mathbb{F}_i \equiv \mathbb{F} \cup \{i\}$. Similarly, $\mathbb{B}_i \equiv \mathbb{B}\backslash\{i\}$ with $\mathbb{B}$ being defined as the complement of $\mathbb{F}$, i.e., $\mathbb{B} \equiv \{1, \ldots, n\}\backslash\mathbb{F}$. Note that the performance of the algorithm improves significantly if one uses (12.17) from Proposition 12.3 instead of (12.12) on lines 7 and 8 in Procedure ASSET-BECOMES-FREE.

CALCULATE-TURNINGPOINTS($\boldsymbol{\mu}, \boldsymbol{\Sigma}, \boldsymbol{l}, \boldsymbol{u}$)

$(\mathbb{F}, \boldsymbol{w}^{(0)}) \leftarrow$ STARTING-SOLUTION($\boldsymbol{\mu}, \boldsymbol{l}, \boldsymbol{u}$)
$\lambda_{\text{current}} \leftarrow \infty$
$t \leftarrow 0$
repeat
  ▷ Case a) Free asset moves to its bound
  ($i\_inside, \lambda_{i\_inside}, b$) $\leftarrow$ ASSET-MOVES-TO-BOUND($\boldsymbol{\mu}, \boldsymbol{\Sigma}, \boldsymbol{l}, \boldsymbol{u}$, $\mathbb{F}, \lambda_{\text{current}}, \boldsymbol{w}^{(t)}$)
  ▷ Case b) Asset on its bound becomes free
  ($i\_outside, \lambda_{i\_outside}$) $\leftarrow$ ASSET-BECOMES-FREE($\boldsymbol{\mu}, \boldsymbol{\Sigma}, \mathbb{F}$, $\lambda_{\text{current}}, \boldsymbol{w}^{(t)}$)

  ▷ Find turning point by comparing cases
  if $i\_inside \neq$ NIL or $i\_outside \neq$ NIL
    then $t \leftarrow t + 1$
      $\boldsymbol{w}_{\boldsymbol{B}}^{(t)} \leftarrow \boldsymbol{w}_{\boldsymbol{B}}^{(t-1)}$
      $\lambda_{\text{current}} \leftarrow \max\{\lambda_{i\_inside}, \lambda_{i\_outside}\}$
      if $\lambda_{i\_inside} = \max\{\lambda_{i\_inside}, \lambda_{i\_outside}\}$
        then
          $\mathbb{F} \leftarrow \mathbb{F} \backslash \{i\_inside\}$
          $w_{i\_inside}^{(t)} \leftarrow b$
        else
          $\mathbb{F} \leftarrow \mathbb{F} \cup \{i\_outside\}$
      $\gamma \leftarrow -\lambda_{\text{current}} \frac{\mathbf{1}_F' \boldsymbol{\Sigma}_F^{-1} \boldsymbol{\mu}_F}{\mathbf{1}_F' \boldsymbol{\Sigma}_F^{-1} \mathbf{1}_F} + \frac{1 - \mathbf{1}_B' \boldsymbol{w}_B^{(t)} + \mathbf{1}_F' \boldsymbol{\Sigma}_F^{-1} \boldsymbol{\Sigma}_{FB} \boldsymbol{w}_B^{(t)}}{\mathbf{1}_F' \boldsymbol{\Sigma}_F^{-1} \mathbf{1}_F}$.
      $\boldsymbol{w}_{\boldsymbol{F}}^{(t)} \leftarrow -\boldsymbol{\Sigma}_{\boldsymbol{F}}^{-1} \boldsymbol{\Sigma}_{\boldsymbol{FB}} \boldsymbol{w}_{\boldsymbol{B}}^{(t)} + \gamma \boldsymbol{\Sigma}_{\boldsymbol{F}}^{-1} \mathbf{1}_F + \lambda_{\text{current}} \boldsymbol{\Sigma}_{\boldsymbol{F}}^{-1} \boldsymbol{\mu}_{\boldsymbol{F}}$
  until $i\_inside =$ NIL and $i\_outside =$ NIL
  ▷ We do not return the starting solution $\boldsymbol{w}^{(0)}$ as it coincides with $\boldsymbol{w}^{(1)}$
  return $\left(\boldsymbol{w}^{(1)}, \boldsymbol{w}^{(2)}, ..., \boldsymbol{w}^{(t)}\right)$

STARTING-SOLUTION($\boldsymbol{\mu}, \boldsymbol{l}, \boldsymbol{u}$)

$\boldsymbol{w} \leftarrow \boldsymbol{l}$
$i \leftarrow \arg\max_j \{\mu_j\}$
while $\mathbf{1}'\boldsymbol{w} < 1$
  do
    $i\_free \leftarrow i$
    $w_i \leftarrow \min\{u_i, l_i + 1 - \mathbf{1}'\boldsymbol{w}\}$
    $i \leftarrow \arg\max_j \{\mu_j | \mu_j < \mu_i\}$
$\mathbb{F} \leftarrow \{i\_free\}$
return $(\mathbb{F}, \boldsymbol{w})$

ASSET-MOVES-TO-BOUND($\boldsymbol{\mu}, \boldsymbol{\Sigma}, \boldsymbol{l}, \boldsymbol{u}, \mathbb{F}, \lambda_{\text{current}}, \boldsymbol{w}$)

```
1  ▷ A sole free asset cannot move to bound
2  if size(𝔽) = 1
3     then
4          return (NIL, NIL, NIL)
5  for i ∈ 𝔽
6       do C_i ← −(1'_F Σ_F^{-1} 1_F)(Σ_F^{-1} μ_F)_i + (1'_F Σ_F^{-1} μ_F)(Σ_F^{-1} 1_F)_i
7          if C_i > 0
8            then b_i ← u_i
9          if C_i < 0
10           then b_i ← l_i
11         λ_i ← Eq. (12.8)
12 i_inside ← arg max_{i∈𝔽}{λ_i | λ_i < λ_current}
13 return (i_inside, λ_{i_inside}, b_{i_inside})
```

ASSET-BECOMES-FREE($\boldsymbol{\mu}, \boldsymbol{\Sigma}, \boldsymbol{l}, \boldsymbol{u}, \mathbb{F}, \lambda_{\text{current}}, \boldsymbol{w}$)

```
1  ▷ Skip procedure if all assets are free
2  if size(𝔽) = n
3     then
4          return (NIL, NIL)
5  for i ∈ 𝔹
6       do 𝔽_i ← 𝔽 ∪ {i}
7          C_i ← −(1'_{F_i} Σ_{F_i}^{-1} 1_{F_i})(Σ_{F_i}^{-1} μ_{F_i})_i + (1'_{F_i} Σ_{F_i}^{-1} μ_{F_i})(Σ_{F_i}^{-1} 1_{F_i})_i
8          λ_i ← Eq. (12.12)
9  i_outside ← arg max_{i∈𝔹}{λ_i | λ_i < λ_current}
10 return (i_outside, λ_{i_outside})
```

Acknowledgment The authors thank Ferenc Niedermayer, William F. Sharpe, Ralph E. Steuer, and Heinz Zimmermann for very helpful comments. We further thank G. Peter Todd for providing us the Excel implementation of the Critical Line Algorithm from Markowitz and Todd (2000).

References

Hirschberger, M., Y. Qi, and R. E. Steuer (2004): *Quadratic Parametric Programming for Portfolio Selection with Random Problem Generation and Computational Experience*, Working papers, Terry College of Business, University of Georgia.

Jorion, P. (1992): "Portfolio Optimization in Practice," *Financial Analysts Journal*, 48(1), 68–74.

Markowitz, H. M. (1952): "Portfolio Selection," *Journal of Finance*, 7(1), 77–91.

Markowitz, H. M. (1956): "The Optimization of a Quadratic Function Subject to Linear Constraints," *Naval Research Logistics Quarterly*, III, 111–133.

Markowitz, H. M. (1959): *Portfolio Selection: Efficient Diversification of Investments*. Wiley, New York, and 1991 2nd ed., Basil Blackwell, Cambridge, MA.

Markowitz, H. M., and P. Todd (2000): *Mean–Variance Analysis in Portfolio Choice and Capital Markets*. Frank J. Fabozzi Associates, New Hope, PA.

Markowitz, H., and N. Usmen (2003): "Resampled Frontiers vs Diffuse Bayes: An Experiment," *Journal Of Investment Management*, 1(4), 9–25.

Michaud, R. (1998): *Efficient Asset Management: A Practical Guide to Stock Portfolio Optimization*. Oxford University Press, Oxford.

Niedermayer, D. (2005): *Portfolio Theory and the Cross-sectional Relation between Expected Returns and Betas*, Master's Thesis, Department of Economics, University of Bern.

Scherer, B. (2002): "Portfolio Resampling: Review and Critique," *Financial Analysts Journal*, 58(6), 98–109.

Scherer, B., and D. R. Martin (2006): *Introduction to Modern Portfolio Optimization with NuOPT, S-PLUS and S+Bayes*, 1st edn. Springer, New York.

Steuer, R. E., Y. Qi, and M. Hirschberger (2006): "Portfolio Optimization: New Capabilities and Future Methods," *Zeitschrift für BWL*, 76(2), 199–219.

Wolf, M. (2006): *Resampling vs. Shrinkage for Benchmarked Managers*, IEW – Working Papers iewwp263, Institute for Empirical Research in Economics – IEW, Available at http://ideas.repec.org/p/zur/iewwpx/263.html

Wolfe, P. (1959): "The Simplex Method for Quadratic Programming," *Econometrica*, 27(3), 382–398.