UNIVERSITÄT BERN

Faculty of Economics and Social Sciences

Department of Economics

Applying Markowitz's Critical Line Algorithm

Andras Niedermayer
Daniel Niedermayer

07-01

March 2006
Revised January 2007

DISCUSSION PAPERS

Schanzeneckstrasse 1
Postfach 8573
CH-3001 Bern, Switzerland
http://www.vwi.unibe.ch

Applying Markowitz's Critical Line Algorithm*

Andras Niedermayer† Daniel Niedermayer‡

March 7, 2006
Revised January 24, 2007

Abstract

We provide a Matlab quadratic optimization tool based on Markowitz's critical line algorithm that significantly outperforms standard software packages and a recently developed operations research algorithm. As an illustration: For a 2000 asset universe our method needs less than a second to compute the whole frontier whereas the quickest competitor needs several hours. This paper can be considered as a didactic alternative to the critical line algorithm such as presented by Markowitz and treats all steps required by the algorithm explicitly. Finally, we present a benchmark of different optimization algorithms' performance.

Keywords: finance, portfolio selection, efficient frontier, critical line algorithm, quadratic optimization, numerical methods
JEL-Classification: C15, C61, C63, G11

*The authors thank Ferenc Niedermayer, William F. Sharpe, 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). This paper has benefited from comments of participants of the Doktorandenkolloquium WWZ Uni Basel (Petersinsel) 2006 and of seminars at the University of Bern.

†Economics Department, University of Bern, Schanzeneckstrasse 1, CH-3001 Bern, Switzerland. Tel.: +41 31 631 39 23. Email: niedermayer@vwi.unibe.ch.

‡WWZ, University of Basel, Holbeinstrasse 12, CH-4051 Basel, Switzerland. Email: daniel.niedermayer@vwi.unibe.ch.

1 Introduction

Markowitz's (1952) Portfolio Theory formulates investors' decisions in a mean-variance setting as a problem of minimizing portfolio variance at a certain level of expected return. The solution set of this problem is visualized by the minimum variance frontier and its positively sloped segment, the *efficient frontier*.

When including the practically relevant condition that short selling cannot take place – thus, that investors cannot weight assets negatively in portfolios – the system of linear equations is extended by n (weak) inequalities, one for each asset. Minimizing portfolio variance with equality and inequality conditions requires computationally expensive quadratic optimization algorithms such as first stated in the critical line algorithm (CLA) of Markowitz (1956) and (1959) and the extended simplex algorithm of Wolfe (1959).

Current research by Steuer, Qi, and Hirschberger (2006) indicates the still remaining need of improving quadratic optimization algorithms' performance.[1] They develop a simplex based algorithm that calculates all turning points of the constrained minimum variance frontier while significantly reducing computational time compared to standard software packages such as Matlab, Cplex, LINGO, Mathematica and premium Solver. When comparing their algorithm's performance with the VBA based implementation of the Optimizer by Markowitz and Todd (2000) they encounter the problem of the 256 column limitation of MS Excel. In order to circumvent this problem we implemented a similar algorithm as in Markowitz and Todd (2000) in

[1] As an example *resampling simulations* as discussed in Michaud (1998) benefit from an algorithmic improvement of quadratic optimization. For a recent application see Wolf (2006).

Fortran 90 where lower and upper bounds on asset weights are imposed.[2] We show that this algorithm outperforms the algorithm in Steuer, Qi, and Hirschberger (2006) by a factor of almost 10 thousand (for 2000 assets) and standard software packages by even more.

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

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: Section 2 introduces the mathematical framework and definitions required by the CLA. Section 3 formulates the quadratic optimization method and the numerical improvement. Section 4 describes performance tests and computational experience. Section 5 concludes.

2 The Framework

Given is a universe of n assets with

$\boldsymbol{\Sigma}$**:** an $(n \times n)$ positive definite covariance matrix,

$\boldsymbol{\mu}$**:** n vector with the assets' expected returns,

[2]The previous version of this paper (Niedermayer and Niedermayer 2006) describes the critical line algorithm with non-negativity constraints.

$\boldsymbol{w}$**:** n vector with the assets' weights.

For a minimum variance portfolio where lower and upper bounds on asset weights are included we define

$\boldsymbol{l}$**:** an n vector containing the asset weights' lower bounds ($w_i \geq l_i, \quad \forall i$),

$\boldsymbol{u}$**:** an n vector containing the asset weights' upper bounds ($w_i \leq u_i, \quad \forall i$),

$\mathbb{F}$**:** a subset of $N = \{1, 2, \ldots, n\}$ containing all assets' indices where weights are within their bounds ($l_i < w_i < u_i$). We shall call the corresponding assets *free assets*.

$\mathbb{B}$**:** 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 $\mathbb{B} = \mathbb{U} \cup \mathbb{L}$.

$k \equiv |\mathbb{F}|$**:** number of elements in $\mathbb{F}$.

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

$$\boldsymbol{\Sigma} = \begin{bmatrix} \boldsymbol{\Sigma_F} & \boldsymbol{\Sigma_{FB}} \\ \boldsymbol{\Sigma_{BF}} & \boldsymbol{\Sigma_B} \end{bmatrix}, \qquad \boldsymbol{\mu} = \begin{bmatrix} \boldsymbol{\mu_F} \\ \boldsymbol{\mu_B} \end{bmatrix} \quad \text{and} \quad \boldsymbol{w} = \begin{bmatrix} \boldsymbol{w_F} \\ \boldsymbol{w_B} \end{bmatrix} \tag{1}$$

with the $(k \times k)$ covariance matrix $\boldsymbol{\Sigma_F}$, the $((n-k) \times (n-k))$ covariance matrix $\boldsymbol{\Sigma_B}$, a $(k \times (n-k))$ matrix $\boldsymbol{\Sigma_{FB}}$, an $((n-k) \times k)$ matrix $\boldsymbol{\Sigma_{BF}}$, 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 $\boldsymbol{\Sigma_{BF}} = \boldsymbol{\Sigma'_{FB}}$.

2.1 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 written as a Lagrange function

$$L = \frac{1}{2}\boldsymbol{w'\Sigma w} - \gamma(\boldsymbol{w'1} - 1) - \lambda(\boldsymbol{w'\mu} - \mu_p)\,, \tag{2}$$

with the Lagrange coefficients γ and λ and the expected return level μ_p.[3] The first constraint in (2) ensures that assets' weights sum to one; the second constraint tells that portfolio variance is minimized at the expected return level of 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 $l_i \leq w_i \leq u_i$.

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 non-negative 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

[3] In the following, we will denote a vector of ones $(1, ..., 1)'$ 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 $\mathbf{1_A}$ (e.g. $\mathbf{1_F}$ has size k and $\mathbf{1_B}$ has size $n-k$.)

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

of the set of solutions in the (μ_p, σ_p) plane as constrained minimum variance frontier (CMVF).

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

Definition 1 A constrained minimum variance portfolio is called *turning point* if in its vicinity other constrained minimum variance portfolios 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.

Proposition 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*

$$\begin{aligned} L =& \frac{1}{2}\boldsymbol{w}_{\boldsymbol{F}}'\boldsymbol{\Sigma}_{\boldsymbol{F}}\boldsymbol{w}_{\boldsymbol{F}} + \frac{1}{2}\boldsymbol{w}_{\boldsymbol{F}}'\boldsymbol{\Sigma}_{\boldsymbol{FB}}\boldsymbol{w}_{\boldsymbol{B}} + \frac{1}{2}\boldsymbol{w}_{\boldsymbol{B}}'\boldsymbol{\Sigma}_{\boldsymbol{BF}}\boldsymbol{w}_{\boldsymbol{F}} + \frac{1}{2}\boldsymbol{w}_{\boldsymbol{B}}'\boldsymbol{\Sigma}_{\boldsymbol{B}}\boldsymbol{w}_{\boldsymbol{B}} \\ & - \gamma(\boldsymbol{w}_{\boldsymbol{F}}'\mathbf{1}_{\boldsymbol{F}} + \boldsymbol{w}_{\boldsymbol{B}}'\mathbf{1}_{\boldsymbol{B}} - 1) - \lambda(\boldsymbol{w}_{\boldsymbol{F}}'\boldsymbol{\mu}_{\boldsymbol{F}} + \boldsymbol{w}_{\boldsymbol{B}}'\boldsymbol{\mu}_{\boldsymbol{B}} - \mu_p)\,, \end{aligned} \tag{3}$$

where the components $\boldsymbol{w}_{\boldsymbol{F}}$ *are subject to minimization and the components* $\boldsymbol{w}_{\boldsymbol{B}}$ *are fixed to their actual values.*

Proof This is obvious: changing $\boldsymbol{w}_{\boldsymbol{F}}$ 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 (2) and (3);[6] the underlying subsets in (3) depend on the constrained minimum variance portfolio's location.

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

[6] Note that for the case $l_i = 0$ and $u_i = \infty$ (3) becomes $L = \frac{1}{2}\boldsymbol{w}_{\boldsymbol{F}}'\boldsymbol{\Sigma}_{\boldsymbol{F}}\boldsymbol{w}_{\boldsymbol{F}} - \gamma(\boldsymbol{w}_{\boldsymbol{F}}'\mathbf{1}_{\boldsymbol{F}} - 1) - \lambda(\boldsymbol{w}_{\boldsymbol{F}}'\boldsymbol{\mu}_{\boldsymbol{F}} - \mu_p)$ which is the same as (2) bar the subscript F.

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 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 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 (3) with respect to $\boldsymbol{w_F}$ yields

$$\boldsymbol{\Sigma_F w_F} + \boldsymbol{\Sigma_{FB} w_B} - \lambda \boldsymbol{\mu_F} = \gamma \mathbf{1_F}. \tag{4}$$

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

$$\mathbf{1'_F w_F} = 1 - \mathbf{1'_B w_B}. \tag{5}$$

Solving (4) together with (5) for γ yields

$$\gamma = -\lambda \frac{\mathbf{1'_F \Sigma_F^{-1} \mu_F}}{\mathbf{1'_F \Sigma_F^{-1} 1_F}} + \frac{1 - \mathbf{1'_B w_B} + \mathbf{1'_F \Sigma_F^{-1} \Sigma_{FB} w_B}}{\mathbf{1'_F \Sigma_F^{-1} 1_F}}. \tag{6}$$

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 (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 $\boldsymbol{\mu' w}$ $(= \mu_p)$ 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 2 with the proof being banned to the appendix.

Proposition 2 *Between two turning points λ and $\boldsymbol{\mu'w}$ are linearly related with a positive slope*

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

□

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 Figure 1.

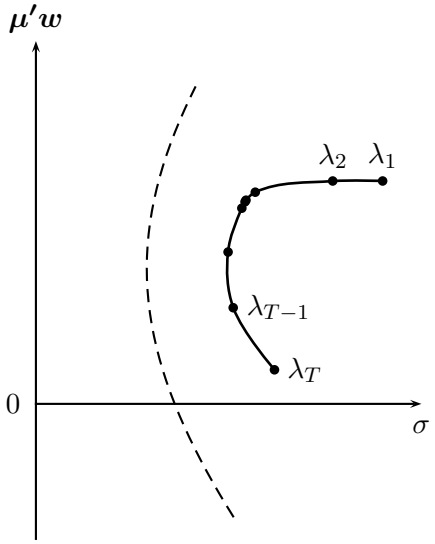

Figure 1: This figure shows the minimum variance frontier (dashed line) and the constrained minimum variance frontier for ten assets and an arbitrarily chosen non-singular covariance matrix. The dots represent the constrained frontier's turning points.

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 (4) will hold for a different subset $\mathbb{F}$. Except for the case where two or more

turning points lie upon each other[7], passing a turning point one has to add or remove exactly one element from the set of free assets $\mathbb{F}$.

Moreover, when moving downwards from a turning point to the next one, λ will decrease (see Proposition 2). When looking at turning points such as in Figure 1 it must therefore be that

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

with T being the number of turning points.

The next two subsections 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. The third subsection shows a way how to improve the algorithm's performance significantly.

3.1 The Starting Solution

This algorithm requires an initial solution on the constrained minimum variance frontier. 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 $\boldsymbol{w'1} < 1$, we start to increase the second assets' weight and so forth. This procedure terminates when $\boldsymbol{w'1} = 1$ is reached.[8]

[7] 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).

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

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 algorithms used in Markowitz and Todd (2000).

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 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 $\boldsymbol{w}$.

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 (4) holds and thus

$$\boldsymbol{w}_{\boldsymbol{F}} = -\boldsymbol{\Sigma}_{\boldsymbol{F}}^{-1}\boldsymbol{\Sigma}_{\boldsymbol{F}\boldsymbol{B}}\boldsymbol{w}_{\boldsymbol{B}} + \gamma\boldsymbol{\Sigma}_{\boldsymbol{F}}^{-1}\mathbf{1}_{\boldsymbol{F}} + \lambda\boldsymbol{\Sigma}_{\boldsymbol{F}}^{-1}\boldsymbol{\mu}_{\boldsymbol{F}}\,. \tag{7}$$

Substituting γ from (6) into (7) gives us $\boldsymbol{w}_{\boldsymbol{F}}$ as a linear function of λ. When decreasing λ the asset i will hit the lower bound if the derivative $dw_i/d\lambda$ is

positive and 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 $\boldsymbol{w_{F}}_i$ and λ resulting from (6) and (7). This gives

$$\begin{aligned}\lambda^{(i)} = \frac{1}{C_i}\Big[\Big(1 - \mathbf{1}_{\boldsymbol{B}}'\boldsymbol{w_B} + \mathbf{1}_{\boldsymbol{F}}'\boldsymbol{\Sigma}_{\boldsymbol{F}}^{-1}\boldsymbol{\Sigma_{FB}w_B}\Big)(\boldsymbol{\Sigma}_{\boldsymbol{F}}^{-1}\mathbf{1_F})_i \\ -(\mathbf{1}_{\boldsymbol{F}}'\boldsymbol{\Sigma}_{\boldsymbol{F}}^{-1}\mathbf{1_F})\Big(b_i + (\boldsymbol{\Sigma}_{\boldsymbol{F}}^{-1}\boldsymbol{\Sigma_{FB}w_B})_i\Big)\Big]\end{aligned} \tag{8}$$

with

$$C_i = -(\mathbf{1}_{\boldsymbol{F}}'\boldsymbol{\Sigma}_{\boldsymbol{F}}^{-1}\mathbf{1_F})(\boldsymbol{\Sigma}_{\boldsymbol{F}}^{-1}\boldsymbol{\mu_F})_i + (\mathbf{1}_{\boldsymbol{F}}'\boldsymbol{\Sigma}_{\boldsymbol{F}}^{-1}\boldsymbol{\mu_F})(\boldsymbol{\Sigma}_{\boldsymbol{F}}^{-1}\mathbf{1_F})_i \tag{9}$$

and[9]

$$b_i = \begin{cases} u_i & \text{if } C_i > 0 \\ l_i & \text{if } C_i < 0. \end{cases} \tag{10}$$

Note that for $k = 1$ one gets $C_i = 0$, which reflects the fact that the constraint $\mathbf{1}_{\boldsymbol{F}}'\boldsymbol{w_F} = w_i = 1 - \mathbf{1}_{\boldsymbol{B}}'\boldsymbol{w_B}$ 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

$$\lambda_{\text{inside}} = \max_{i \in \mathbb{F}}\{\lambda^{(i)}\}, \tag{11}$$

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

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).

[9] Note that $dw_i/d\lambda = -C_i/(\mathbf{1}_{\boldsymbol{F}}'\boldsymbol{\Sigma}_{\boldsymbol{F}}^{-1}\mathbf{1_F})$. 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).

Case b) one asset formerly on its bound wants to become free

When moving downwards in $\boldsymbol{\mu'w}$ 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

$$\mathbb{F}_i \equiv \mathbb{F} \cup \{i\}$$
$$\mathbb{B}_i \equiv \mathbb{B} \setminus \{i\},$$

where $i \in \mathbb{B}$. Analogously to (8) the value $\lambda^{(i)}$ where the newly included asset i's weight moves away from its bound is given by

$$\begin{aligned}\lambda^{(i)} = \frac{1}{C_i}\Big[\Big(1 - \mathbf{1'_{B_i}} \boldsymbol{w_{B_i}} + \mathbf{1'_{F_i}} \boldsymbol{\Sigma}_{\boldsymbol{F_i}}^{-\mathbf{1}} \boldsymbol{\Sigma_{F_i B_i}} \boldsymbol{w_{B_i}}\Big)(\boldsymbol{\Sigma}_{\boldsymbol{F_i}}^{-\mathbf{1}} \mathbf{1_{F_i}})_i \\ -(\mathbf{1'_{F_i}} \boldsymbol{\Sigma_{F_i}}^{-\mathbf{1}} \mathbf{1_{F_i}})\Big(b_i + (\boldsymbol{\Sigma}_{\boldsymbol{F_i}}^{-\mathbf{1}} \boldsymbol{\Sigma_{F_i B_i}} \boldsymbol{w_{B_i}})_i\Big)\Big]\end{aligned} \tag{12}$$

with

$$C_i = -(\mathbf{1'_{F_i}} \boldsymbol{\Sigma}_{\boldsymbol{F_i}}^{-\mathbf{1}} \mathbf{1_{F_i}})(\boldsymbol{\Sigma}_{\boldsymbol{F_i}}^{-\mathbf{1}} \boldsymbol{\mu_{F_i}})_i + (\mathbf{1'_{F_i}} \boldsymbol{\Sigma}_{\boldsymbol{F_i}}^{-\mathbf{1}} \boldsymbol{\mu_{F_i}})(\boldsymbol{\Sigma}_{\boldsymbol{F_i}}^{-\mathbf{1}} \mathbf{1_{F_i}})_i \,. \tag{13}$$

and where for convenience b_i is used for $(\boldsymbol{w_{F_i}})_i = w_i$, 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, if it was on the lower bound it stands for l_i.[10]

In order to find the maximal $\lambda^{(i)} < \lambda_{\text{current}}$ where an asset i currently on its bound wants to become free (12) must be applied for all $i \in \mathbb{B}$.

$$\lambda_{\text{outside}} = \max_{i \in \mathbb{B}} \{\lambda^{(i)} \mid \lambda^{(i)} < \lambda_{\text{current}}\}\,. \tag{14}$$

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

[10] 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 $dw_i/d\lambda$ 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.

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.[11]

The pseudo-code in Appendix B summarizes the algorithm for the simplified case when $l_i = 0$ and $u_i = +\infty$.[12]

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.

[11] 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 $\lambda = 0$.

[12] 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.

Note that in contrast to the calculations in (8) the specification of $\mathbb{F}_i$ in (12) depends on i and $\boldsymbol{\Sigma}_{\boldsymbol{F_i}}^{-\mathbf{1}}$ must be recalculated for each $i \notin \mathbb{F}$.[13] 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.

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.

Expansion of the covariance matrix $\boldsymbol{\Sigma}_{\boldsymbol{F}}$

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

$$\begin{bmatrix} \boldsymbol{A} & \boldsymbol{a} \\ \boldsymbol{a'} & \alpha \end{bmatrix}^{-1} = \begin{bmatrix} \boldsymbol{A^{-1}} + \beta \boldsymbol{c}\boldsymbol{c'} & -\beta \boldsymbol{c} \\ -\beta \boldsymbol{c'} & \beta \end{bmatrix} \tag{15}$$

holds where

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

Proof Multiplying the expanded matrix with the right-hand side of (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, $\boldsymbol{\Sigma}_{\boldsymbol{F_i}}^{-\mathbf{1}}$, for the new subset. Lemma 1 frees us from the burden of making this calculation all

[13] 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.

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 $2k^2$.

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 2 where for presentational purposes the given index is assumed to be the last one.

Lemma 2 *Let $\boldsymbol{A}$ and $\boldsymbol{B}$ be $k \times k$ matrices, $\boldsymbol{a}$ and $\boldsymbol{b}$ k vectors and α and β two scalars. Then if*

$$\begin{bmatrix} \boldsymbol{A} & \boldsymbol{a} \\ \boldsymbol{a'} & \alpha \end{bmatrix}^{-1} = \begin{bmatrix} \boldsymbol{B} & \boldsymbol{b} \\ \boldsymbol{b'} & \beta \end{bmatrix} \tag{16}$$

holds then

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

holds as well.

PROOF By combining (15) and (16) and solving for $\boldsymbol{A^{-1}}$. ∎

3.3.1 Key Improvement

The most remarkable improvement stems from the fact that in (12) we need to know neither $\boldsymbol{\Sigma_{F_i}^{-1}}$ nor $\boldsymbol{\Sigma_{F_i}^{-1}\Sigma_{F_iB_i}}$. This is stated in Proposition 3.

Proposition 3 *Expression* (12),

$$\lambda^{(i)} = \frac{1}{C_i}\Big[\Big(1 - \boldsymbol{1'_{B_i}w_{B_i}} + \boldsymbol{1'_{F_i}\Sigma_{F_i}^{-1}\Sigma_{F_iB_i}w_{B_i}}\Big)(\boldsymbol{\Sigma_{F_i}^{-1}1_{F_i}})_i - (\boldsymbol{1'_{F_i}\Sigma_{F_i}^{-1}1_{F_i}})\Big(b_i + (\boldsymbol{\Sigma_{F_i}^{-1}\Sigma_{F_iB_i}w_{B_i}})_i\Big)\Big]$$

with

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

can be rewritten as

$$\begin{aligned} \lambda^{(i)} = \frac{1}{D_i} \Big[\Big(1 - \mathbf{1}'_{\boldsymbol{B}} \boldsymbol{w}_{\boldsymbol{B}} + \mathbf{1}'_{\boldsymbol{F}} \boldsymbol{\Sigma}_{\boldsymbol{F}}^{-1} \boldsymbol{\Sigma}_{\boldsymbol{FB}} \boldsymbol{w}_{\boldsymbol{B}} \Big) (1 - \boldsymbol{a}' \boldsymbol{\Sigma}_{\boldsymbol{F}}^{-1} \mathbf{1}_{\boldsymbol{F}}) \\ + \Big(\boldsymbol{a}' \boldsymbol{\Sigma}_{\boldsymbol{F}}^{-1} \boldsymbol{\Sigma}_{\boldsymbol{FB}} \boldsymbol{w}_{\boldsymbol{B}} - \boldsymbol{\Sigma}_{\boldsymbol{iB}} \boldsymbol{w}_{\boldsymbol{B}} \Big) (\mathbf{1}'_{\boldsymbol{F}} \boldsymbol{\Sigma}_{\boldsymbol{F}}^{-1} \mathbf{1}_{\boldsymbol{F}}) \Big] \end{aligned} \tag{17}$$

where the used variables are defined as

$$\boldsymbol{\Sigma}_{\boldsymbol{F_i}} = \begin{bmatrix} \boldsymbol{\Sigma}_{\boldsymbol{F}} & \boldsymbol{a} \\ \boldsymbol{a}' & \alpha \end{bmatrix}, \qquad \boldsymbol{\mu}_{\boldsymbol{F_i}} = \begin{bmatrix} \boldsymbol{\mu}_{\boldsymbol{F}} \\ \mu_i \end{bmatrix}$$

and

$$D_i = (1 - \boldsymbol{a}' \boldsymbol{\Sigma}_{\boldsymbol{F}}^{-1} \mathbf{1}_{\boldsymbol{F}})(\mathbf{1}'_{\boldsymbol{F}} \boldsymbol{\Sigma}_{\boldsymbol{F}}^{-1} \boldsymbol{\mu}_{\boldsymbol{F}}) - (\mu_i - \boldsymbol{a}' \boldsymbol{\Sigma}_{\boldsymbol{F}}^{-1} \boldsymbol{\mu}_{\boldsymbol{F}})(\mathbf{1}'_{\boldsymbol{F}} \boldsymbol{\Sigma}_{\boldsymbol{F}}^{-1} \mathbf{1}_{\boldsymbol{F}}).$$

PROOF The derivation is shown in the appendix. ■

It is remarkable that in (17) the vector $\boldsymbol{a}$ (the ith column of Σ corresponding to a trial $i \in \mathbb{B}$) enters linearly. Therefore the calculation of $\lambda^{(i)}$ is fast, one should calculate only two scalar products with $\boldsymbol{a}$ for each $i \in \mathbb{B}$.

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 comparing CPU times directly.

However, there are two reasons for us to make such performance comparisons. 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, Qi, and Hirschberger (2006)[14], whose simplex based multi-parametric optimization algorithm was implemented in Java (Java MPQ). The latter comparison is important; as argued in Steuer, Qi, and Hirschberger (2006), the MPQ outperforms Matlab, Cplex, LINGO, Mathematica, and Excel's premium Solver. Steuer, Qi, and Hirschberger (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

[14]Steuer, Qi, and Hirschberger (2006) run their tests on a Dell 3.06 GHz desktop as well.

the other problems, it can solve the optimization problem for a more general set of constraints.

For the tests illustrated in Figure 2 a positive definite covariance matrix was generated as

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

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, Qi, and Hirschberger (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 (8) and (12), we made sure that $\hat{k}/n$ is similar to that in Steuer, Qi, and Hirschberger (2006). In our tests with 1000 assets $\hat{k}$ was 60 and when testing with 2000 assets $\hat{k}$ was 250.

In Figure 2 both axes are logarithmic. The slope of the linear fit (of an 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 1.

Since the Wolfe-Simplex algorithm and the Matlab quadratic optimization package only calculate one single point on the constrained minimum variance frontier 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, Qi, and Hirschberger (2006), the CPU times reported in Table 1 support our method even more.

As in Steuer, Qi, and Hirschberger (2006), our tests were conducted on a Dell desktop with a 3.06 GHz processor.

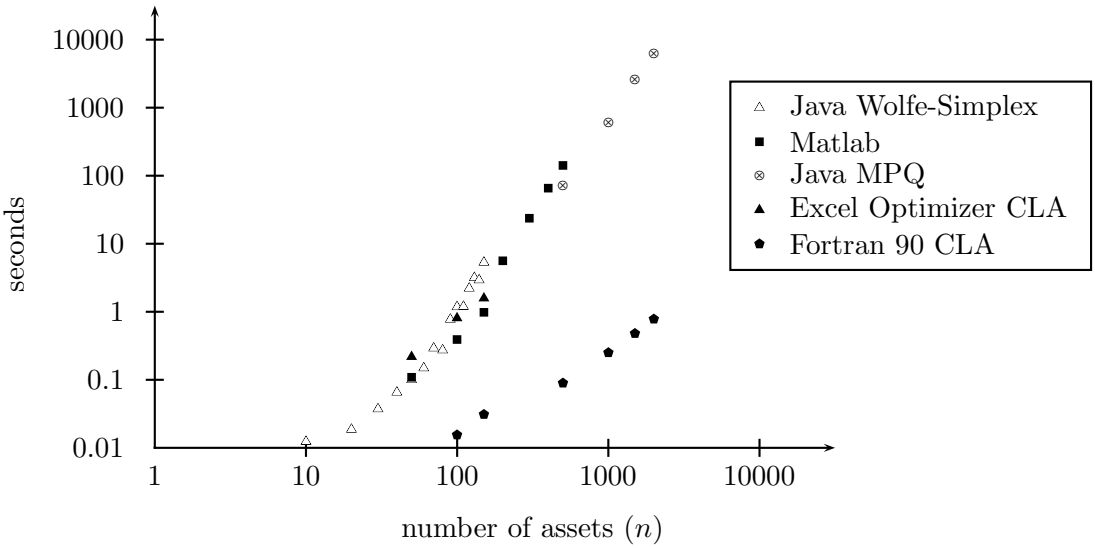

Figure 2: Testing different algorithms for the case with lower bounds zero and upper bounds infinity: CPU times for different number of assets and randomly generated positive definite covariance matrix.

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	-
1000	0.25	-	-	602	-
1500	0.48	-	-	2580	-
2000	0.78	-	-	6300	-
perf.	$\mathcal{O}(n^{1.6})$	$\mathcal{O}(n^{3.6})$	$\mathcal{O}(n^{3.2})$	$\mathcal{O}(n^{3.3})$	$\mathcal{O}(n^{1.8})$

Table 1: Different CPU times in seconds for the case with lower bounds zero and upper bounds infinity. 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, Qi, and Steuer (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 $\mathcal{O}(n^{1.3})$.

5 Conclusions

This paper presents the critical line algorithm (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 CLA – available on request from the authors in form of a Matlab package – outperforms the current Matlab optimization tool by a factor of approximately 15 thousand when the problem size (number of assets) is 2000. When comparing with the algorithm in Steuer, Qi, and Hirschberger (2006) that also computes all turning points analytically such as the CLA does, the performance improvement is still around 8 thousand.

As all steps of the algorithm are treated explicitly, this code can be di-

rectly used for the implementation in other programming languages and used for problems of large scale portfolio optimization and CPU time intensive Monte Carlo simulations.

Appendix

A Proofs

PROOF (PROPOSITION 2) For tractability we define three constants

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

From equation (4) and the definitions given in (1) follows that

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

Differentiating this expression with respect to λ yields

$$\frac{\partial(\boldsymbol{\mu}' \boldsymbol{w})}{\partial \lambda} = C_{\mu\mu} - \frac{C_{1\mu}^2}{C_{11}}\,. \tag{18}$$

Since between two turning points $\boldsymbol{\Sigma}_{\boldsymbol{F}}$ does not change, $\mu_p(\lambda) = \boldsymbol{\mu}' \boldsymbol{w}(\lambda)$ is linear in λ with a slope given by (18). We show below that this slope is positive.

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

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

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

Positive definiteness of $\boldsymbol{\Sigma}_{\boldsymbol{F}}^{-1}$ means $\boldsymbol{x}' \boldsymbol{\Sigma}_{\boldsymbol{F}}^{-1} \boldsymbol{x} > 0$ 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. Therefore, the discriminant is negative which gives

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

■

PROOF (PROPOSITION 3) According to Lemma 1 $\boldsymbol{\Sigma}_{\boldsymbol{F_i}}^{-\mathbf{1}}$ can be expressed in terms of $\boldsymbol{\Sigma}_{\boldsymbol{F}}^{-\mathbf{1}}$, $\boldsymbol{a}$ and α. Multiplying $\boldsymbol{\Sigma}_{\boldsymbol{F_i}}^{-\mathbf{1}}$ by $\mathbf{1}_{\boldsymbol{F_i}}$ and $\boldsymbol{\mu}_{\boldsymbol{F_i}}$ respectively yields

$$\boldsymbol{\Sigma}_{\boldsymbol{F_i}}^{-\mathbf{1}}\mathbf{1}_{\boldsymbol{F_i}} = \begin{bmatrix} \boldsymbol{\Sigma}_{\boldsymbol{F}}^{-\mathbf{1}}\mathbf{1}_{\boldsymbol{F}} - \beta(1 - \boldsymbol{c}'\mathbf{1})\boldsymbol{c} \\ \beta(1 - \boldsymbol{c}'\mathbf{1}) \end{bmatrix} \tag{19}$$

and

$$\boldsymbol{\Sigma}_{\boldsymbol{F_i}}^{-\mathbf{1}}\boldsymbol{\mu}_{\boldsymbol{F_i}} = \begin{bmatrix} \boldsymbol{\Sigma}_{\boldsymbol{F}}^{-\mathbf{1}}\boldsymbol{\mu}_{\boldsymbol{F}} - \beta(\mu_i - \boldsymbol{c}'\boldsymbol{\mu}_{\boldsymbol{F}})\boldsymbol{c} \\ \beta(\mu_i - \boldsymbol{c}'\boldsymbol{\mu}_{\boldsymbol{F}}) \end{bmatrix}. \tag{20}$$

Multiplying (19) and (20) by $\mathbf{1}'_{\boldsymbol{F_i}}$ and plugging the values into (12) yields the denominator in (17).

Using the following properties and Lemma 1 yields the numerator in (17):

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

■

B Pseudo-Code

The following pseudo-code describes the procedure which calculates all turning points of the minimum variance frontier where restrictions on negative portfolio weights are imposed. Parameters are the covariance matrix $\boldsymbol{\Sigma}$ and expected returns $\boldsymbol{\mu}$. Asset weights $\boldsymbol{w}_{\boldsymbol{F}}^{(t)}$ are returned for each turning point t. Between turning points weights are linear combinations of the two surrounding turning points. All equation numbers in the pseudo-code and its description below refer to equations in Niedermayer and Niedermayer (2006).

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

$j \leftarrow \arg\max_i \mu_i$
$w_j^{(1)} \leftarrow 1$; $\forall i \neq j : w_i^{(1)} \leftarrow 0$
$\mathbb{F} \leftarrow \{j\}$
$\lambda_{\text{current}} \leftarrow \infty$
$t \leftarrow 1$
repeat
  ▷ Case a) Free asset moves to its bound
  if $size(\mathbb{F}) > 1$
    then
      for $i \in \mathbb{F}$
        do $C_i \leftarrow -(\mathbf{1}'_{\boldsymbol{F}}\boldsymbol{\Sigma}_{\boldsymbol{F}}^{-1}\mathbf{1}_{\boldsymbol{F}})(\boldsymbol{\Sigma}_{\boldsymbol{F}}^{-1}\boldsymbol{\mu}_{\boldsymbol{F}})_i + (\mathbf{1}'_{\boldsymbol{F}}\boldsymbol{\Sigma}_{\boldsymbol{F}}^{-1}\boldsymbol{\mu}_{\boldsymbol{F}})(\boldsymbol{\Sigma}_{\boldsymbol{F}}^{-1}\mathbf{1}_{\boldsymbol{F}})_i$
          $\lambda_i \leftarrow -(\boldsymbol{\Sigma}_{\boldsymbol{F}}^{-1}\mathbf{1}_{\boldsymbol{F}})_i / C_i$
                                                ▷ Eq. (9)
      $i\_inside \leftarrow \arg\max_{i\in\mathbb{F}}\{\lambda_i | \lambda_i < \lambda_{\text{current}}\}$
  ▷ Case b) Asset on its bound becomes free
  if $size(\mathbb{F}) < size(\boldsymbol{\mu})$
    then
      for $i \notin \mathbb{F}$
        do $\mathbb{F}_i \leftarrow \mathbb{F} \cup \{i\}$
          $C_i \leftarrow -(\mathbf{1}'_{\boldsymbol{F_i}}\boldsymbol{\Sigma}_{\boldsymbol{F_i}}^{-1}\mathbf{1}_{\boldsymbol{F_i}})(\boldsymbol{\Sigma}_{\boldsymbol{F_i}}^{-1}\boldsymbol{\mu}_{\boldsymbol{F_i}})_i + (\mathbf{1}'_{\boldsymbol{F_i}}\boldsymbol{\Sigma}_{\boldsymbol{F_i}}^{-1}\boldsymbol{\mu}_{\boldsymbol{F_i}})(\boldsymbol{\Sigma}_{\boldsymbol{F_i}}^{-1}\mathbf{1}_{\boldsymbol{F_i}})_i$
          $\lambda_i \leftarrow (\boldsymbol{\Sigma}_{\boldsymbol{F_i}}^{-1}\mathbf{1}_{\boldsymbol{F_i}})_i / C_i$
                                                ▷ Eq. (11)
      $i\_outside \leftarrow \arg\max_{i\notin\mathbb{F}}\{\lambda_i | \lambda_i < \lambda_{\text{current}}\}$
  ▷ Find turning point by comparing cases
  if $i\_inside \neq$ NIL or $i\_outside \neq$ NIL
    then $t \leftarrow t + 1$
      $\lambda_{\text{current}} \leftarrow \max\{\lambda_{i\_inside}, \lambda_{i\_outside}\}$
      if $\lambda_{i\_inside} = \max\{\lambda_{i\_inside}, \lambda_{i\_outside}\}$
        then $\mathbb{F} \leftarrow \mathbb{F}\backslash\{i\_inside\}$
        else $\mathbb{F} \leftarrow \mathbb{F} \cup \{i\_outside\}$
      $\gamma \leftarrow \frac{1}{\mathbf{1}'_{\boldsymbol{F}}\boldsymbol{\Sigma}_{\boldsymbol{F}}^{-1}\mathbf{1}_{\boldsymbol{F}}} - \frac{\mathbf{1}'_{\boldsymbol{F}}\boldsymbol{\Sigma}_{\boldsymbol{F}}^{-1}\boldsymbol{\mu}_{\boldsymbol{F}}}{\mathbf{1}'_{\boldsymbol{F}}\boldsymbol{\Sigma}_{\boldsymbol{F}}^{-1}\mathbf{1}_{\boldsymbol{F}}}\lambda_{\text{current}}$ ▷ Eq. (4)
      $\boldsymbol{w}_{\boldsymbol{F}}^{(t)} \leftarrow \lambda_{\text{current}}\boldsymbol{\Sigma}_{\boldsymbol{F}}^{-1}\boldsymbol{\mu}_{\boldsymbol{F}} + \gamma\boldsymbol{\Sigma}_{\boldsymbol{F}}^{-1}\mathbf{1}_{\boldsymbol{F}}$ ▷ Eq. (7)
until $i\_inside =$ NIL and $i\_outside =$ NIL
return $\left\{\boldsymbol{w}_{\boldsymbol{F}}^{(1)}, \boldsymbol{w}_{\boldsymbol{F}}^{(2)}, ..., \boldsymbol{w}_{\boldsymbol{F}}^{(t)}\right\}$

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 $\boldsymbol{\Sigma_F}$ 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 $\boldsymbol{\Sigma_{F_i}}$ and $\boldsymbol{\mu_{F_i}}$ with $\mathbb{F}_i \equiv \mathbb{F} \cup \{i\}$. Note that the performance of the algorithms improves significantly if one uses (16) from Proposition 3 instead of (11) on line 20.

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.

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

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

——— (1959): *Portfolio Selection: Efficient Diversification of Investments*. John Wiley and Sons, 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, Pennsylvania.

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

Niedermayer, A., and D. Niedermayer (2006): "Applying Markowitz's Critical Line Algorithm," Diskussionsschriften dp0602, Universitat Bern, Volkswirtschaftliches Institut.

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

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

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.