Eidgenössische Technische Hochschule Zürich
Swiss Federal Institute of Technology Zurich

Portfolio Optimization for Constrained Shortfall Risk: Implementation and IT Architecture Considerations

Matthias Kull

ETH Zürich

Master Thesis

Portfolio Optimization for Constrained Shortfall Risk: Implementation and IT Architecture Considerations

Author: Matthias Kull

Examiner: Prof. Dr. Didier Sornette, ETH Zürich

Advisors: Dr. Donnacha Daly, ETH Zürich
Zalán Forró, ETH Zürich
Dr. Michael Markovich, Credit Suisse

A thesis submitted in fulfilment of the requirements for the degree of Master of Advanced Studies

on the chair of

Entrepreneurial Risks

at the department of

Management, Technology and Economics (MTEC)

Zürich, July 2014

Declaration of Originality

I hereby confirm that I am the sole author of the written work entitled 'Portfolio Optimization for Constrained Shortfall Risk: Implementation and IT Architecture Considerations' and that I have compiled it in my own words. Parts excepted are corrections of form and content by the supervisor. With my signature I confirm that I have

- committed none of the forms of plagiarism described in the 'Citation etiquette' information sheet *plagiarism-citationetiquette.pdf*,
- documented all methods, data and processes truthfully,
- not manipulated any data, and
- mentioned all persons who were significant facilitators of the work.

I am aware that the work may be screened electronically for plagiarism.

Zürich, July 2014

Matthias Kull

Abstract

The efficient optimization of portfolios with a large number of instruments and scenarios as well as the adequate modeling of utility functions, risks, and constraints are the basic requirements of portfolio optimization. The development of optimization techniques has come a long way. Traditional portfolio theory lacks on the assumption of normal distributed returns, which is not the case in financial markets, and current finance regulations require a risk management that focuses on tail risk, which is difficult to handle in portfolio optimization. An approach of Rockafellar and Uryasev that uses a technique, where Value-at-Risk and Conditional Value-at-Risk are optimized simultaneously, overcomes these issues. The current thesis elaborates the portfolio optimization approach of Rockafellar and Uryasev step by step and describes how it can be implemented with linear programming. The implemented approach is tested in Matlab with a portfolio composed of different asset classes to demonstrate how the optimization works in practice. In addition, an IT architecture is proposed to efficiently manage the simultaneous optimization of large portfolios with a large number of scenarios in a multi-user environment to support a sophisticated investment process.

Contents

Chapter 1

Introduction

1.1 Background

The development of optimization approaches is of major concern for quantitative research departments. Their aim is to provide an optimization approach, which is broadly accepted and used in the portfolio management business, in order to allocate portfolios according to the specified risk. Therefore, the risks of financial markets need to be estimated properly and controlled to provide reliable and stable solutions. The best-known portfolio optimization approach was introduced by Markowitz [16] when he published his seminar paper about Portfolio Selection in 1952. However, his approach has important downsides: It uses volatility to measure risk which is only ideal for normal distributions and it penalizes losses equally to returns of the same magnitude. The recent past shows that financial markets, especially during crises, do not follow the logic of normal distributed returns, which results in a misinterpretation of the underlying portfolio risks.

Nowadays, the trend is to use downside risk measures such as Value-at-Risk[1] (VaR) or Conditional Value-at-Risk[2] (CVaR) for portfolio optimization because finance regulations, such as Basel III[3], formulate some of the risk management requirements in terms of percentiles of return distributions. However, due to the characteristics of such downside risk measures, optimization approaches based on these measures are not very convenient to use, do not scale well, and can also lead to sub-optimal portfolio allocations.

[1] Value-at-Risk is the worst loss over a certain period of time which will not be exceeded with a certain level of confidence [14].

[2] Conditional Value-at-Risk is the average loss given that the loss is greater than VaR with a certain level of confidence [20].

[3] Basel III is a global, voluntary regulatory standard on bank capital adequacy, stress testing and market liquidity risk. It was supposed to strengthen bank capital requirements by increasing bank liquidity and decreasing bank leverage.

1.2 Aim and Purpose

The aim of the present thesis is to provide an implementation guide to optimize portfolios for constrained shortfall risk as well as a proposal for an IT architecture, which is able to efficiently manage the simultaneous optimization of very large portfolios in a multi-user environment, integrated in an established investment process. The task was initiated by the team of Quantitative Research at the department of Private Banking and Wealth Management of Credit Suisse. They requested a portfolio optimization approach to minimize shortfall risk either by an analytical or an empirical method.

1.3 Methodology

The current thesis is mainly based on the optimization approach of Rockafellar and Uryasev [20] which focuses on minimizing Conditional Value-at-Risk. Central to their approach is that they derived a function for CVaR which is independent of the VaR function. This means that VaR does not have to be calculated first, but it will be calculated as side-effect of the CVaR optimization. Another major benefit of the approach is that it can be used for any distribution and that it can be implemented by linear programming. This stimulates the two basic requirements of portfolio optimization: (1) adequate modeling of utility functions, risks and constraints and (2) the ability to handle large number of instruments and scenarios.

Firstly, concepts and methods of traditional portfolio theory are analyzed. Therefore, the distributions of returns in financial markets are studied as well as volatility and VaR in the context of a coherent risk measure framework, and their issues for optimization techniques are clarified. Secondly, the portfolio optimization approach of Rockafellar and Uryasev [20] is elaborated and implemented in Matlab with linear programming techniques as proof of concept. The implementation is used to test the approach as well as to analyze the results on a portfolio composed of different asset classes. Thirdly, an IT architecture illustrates the integration of the elaborated approach of Rockafellar and Uryasev [20] into an established investment process of a bank. Thus, the corresponding portfolio management process is described. It is demonstrated how the target system is used by different user roles to represent the defined process of portfolio management with mean-variance as well as with the optimization approach of Rockafellar and Uryasev [20]. Finally, the functionality of the system components and the required interfaces are highlighted.

1.4 Scope

The major focus of the present thesis is on the core concept of the approach of Rockafellar and Uryasev [20]. It is shown how it can be implemented with linear programming to optimize portfolios with constraints on CVaR as well as lower and upper bounds to define minimum and maximum weights for the underlying assets in the portfolio. The approach is also designed to have additional constraints for transaction costs and liquidity. However, these constraints are not part of the implementation in the current thesis. The IT architecture is considered to be a concept of how the optimization approach can be applied in an IT system landscape. There is no analysis about the performance of such a system in the scope of the present thesis.

1.5 Target Audience

Due to the methodology based on a theoretical background with focus on a practical solution, the target audience is:

- Quantitative Analysts
- Risk Managers
- Portfolio Managers
- IT Architects and Engineers
- Students with basic knowledge in statistics and mathematical finance

1.6 Thesis Structure

The thesis is structured into the following seven chapters:

- *Chapter 1* describes the background of portfolio optimization, introduces the used methodology and the aim and purpose of the thesis. In addition, the scope as well as the target audience are defined.
- *Chapter 2* presents an overview of the core concepts of portfolio theory, and answers the question of which properties are needed to fulfill the requirements for a coherent risk measure. It also contains a description of volatility and illustrates its limitations. Furthermore, downside risk measures are introduced and it is discussed why CVaR is the preferred risk measure for portfolio optimization. Finally, the basic concepts of portfolio optimization are explained, and the limitations of the mean-variance approach are discussed.

- *Chapter 3* is the main chapter of the thesis. It contains the elaboration of the Rockafellar and Uryasev approach to optimize portfolios with CVaR constraints and how it can be implemented with linear programming.
- *Chapter 4* presents two methods to simulate correlated future asset prices.
- *Chapter 5* contains the results of the implemented optimization approach, tested with a portfolio of different asset classes. The two different methods of Chapter 4 are used to simulate scenarios.
- *Chapter 6* describes an IT architecture inside an investment process, which supports the portfolio management to use traditional mean-variance optimization and the optimization with constrained shortfall risk.
- *Chapter 7* concludes the present thesis by emphasizing important findings and provides an outlook for further research of this topic.

Chapter 2

Portfolio Theory

This chapter provides an overview of the theory behind returns, their distributions, and the concept of Markowitz' mean-variance portfolio optimization. In addition, we look at the limitations of volatility as risk measure, the issues of Value-at-Risk, and the advantages of Conditional Value-at-Risk for portfolio optimization.

2.1 Assets and Portfolios

A financial asset is an intangible asset that represents a claim on future cash flows. Another term used for a financial asset is financial instrument. It is often referred to certain types of financial instruments as securities. For every financial instrument there is a minimum of two parties. The party that has agreed to make future cash payments is called the issuer; the party that owns the financial instrument and, therefore, the right to receive the payments made by the issuer is called the investor [7]. Typically, financial assets are traded on financial markets. Examples of financial assets are:

- Equities
- Bonds (e.g., corporate bonds, government bonds)
- Cash (e.g., US treasury bills)
- Derivatives (e.g., options, futures, forwards, swaps)
- Funds (e.g., mutual funds, hedge funds, ETF)

A portfolio can be defined by positions with a certain number of constituent assets [14].

2.2 Return

In the context of portfolio theory, a return is a gain of a financial asset or a portfolio in a particular period. Thus, a loss within a particular period is a negative return for the same time range. Returns and losses are usually quoted as a percentage. As a general rule, one can say that the more risk someone takes, the greater the potential for a higher return, but also for a higher loss. The definition for a return R_t over a single period t is

$$R_t = \frac{P_t - P_0}{P_0}, \tag{2.1}$$

where

- P_t is the price of the asset at the end of the period, and
- P_0 is the price of the asset at the begin of the period.

Expressed in logarithm terms, the return R_t can be described as

$$R_{t(log)} = \log(\frac{P_t}{P_0}). \tag{2.2}$$

2.2.1 Expected Return

The expected return defines how big or small the return or the loss will be in the future. The best unbiased estimate for the expected return $E(R)$ under the efficient market hypothesis is the mean of the historical returns

$$E(R) = \frac{1}{N}\sum_{i=1}^{N} R_i, \tag{2.3}$$

where R_i is the return for a specific time and N is the number of time steps (e.g., trading days). The expected return on a portfolio of assets $E(R_p)$ over a specific time period is a weighted average of the returns on the individual assets in the portfolio [4, 7], and can be calculated in the following way:

$$E(R_p) = w_1 * R_1 + w_2 * R_2 + \ldots + w_N * R_N, \tag{2.4}$$

where $w_1, \ldots, w_N$ are the asset weights in the portfolio and $R_1, \ldots, R_N$ are the expected rates of return for the different assets.

As an example for a simple portfolio of two assets, the first asset is a stock of a company and the second is a government bond. If we expect the stock asset to return 10% and the bond asset to return 6%, and the allocation to the stock asset is 30% and to the bond asset 70%, we have an expected return of the portfolio of 7.2% calculated with the following formula:

$$E(R_p) = (0.3) * (0.1) + (0.7) * (0.06) = 0.072 \tag{2.5}$$

The expected return does not guarantee a rate of return. However, it is an established forecast to estimate the future value of portfolios and, in addition, it provides a measure of actual returns.

2.2.2 Distribution of Returns

2.2.2.1 Normal Distributions

As shown in Figure 2.1 and Figure 2.2, the normal distribution is a symmetric distribution where outcomes above and below the expected value are equally likely [7]. The corresponding probability density function is defined as

$$\phi(x) = \frac{1}{\sigma\sqrt{2\pi}} e^{-\frac{(x-\mu)^2}{2\sigma^2}}. \tag{2.6}$$

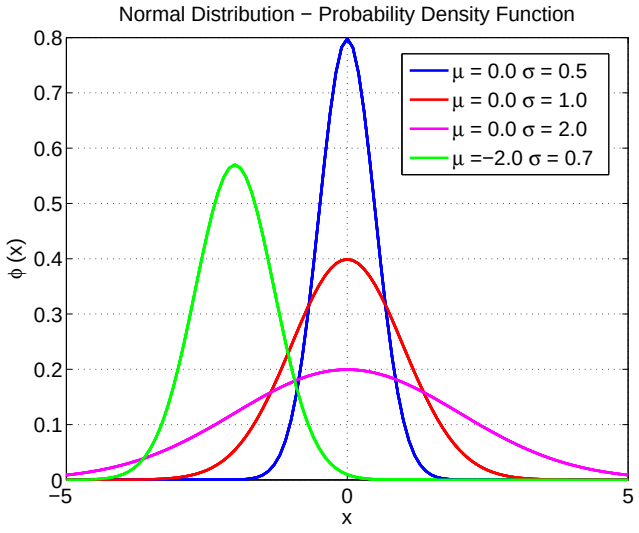

Figure 2.1: Probability density function (PDF) for the normal distribution with different values for mean μ and standard deviation σ. Due to the symmetry of the distribution, outcomes above and below the mean μ are equally likely.

The cumulative distribution function of the normal distribution is the integral

$$\Phi(x) = \frac{1}{\sqrt{2\pi}} \int_{-\infty}^{x} e^{-t^2/2} \, \mathrm{d}t. \tag{2.7}$$

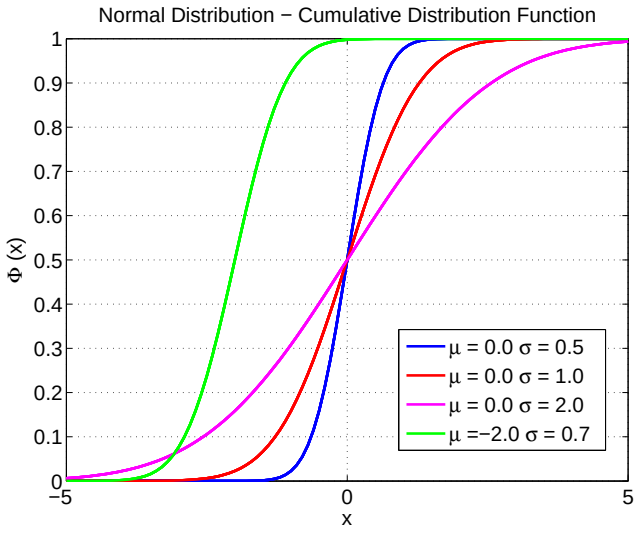

Figure 2.2: Cumulative distribution function (CDF) for the normal distribution with different values for mean μ and standard deviation σ.

There are empirical studies of real-world financial markets as well as theoretical arguments that argue that the normal distribution as a model for asset returns is not a good assumption[4].

2.2.2.2 Real World Distributions

In the real world, many non-normal distributions can be observed. These distributions are characterized by a skewness and a kurtosis different from 0. Skewness is a measure of symmetry, more precisely, the greater the skewness the greater the lack of symmetry [6]. It is known as the third empirical moment. As shown in Figure 2.3, negative skews (skewed left) have a longer left tail and the mass of the distribution is concentrated on the right of the figure. On the other other hand positive skews (skewed right) have a longer right tail and the mass of the distribution is concentrated on the left of the figure. Investors prefer positive skews – they dislike negative returns more than they like the same level of positive returns. The skewness sk is defined as

$$sk = \frac{1}{N-1} \sum_{i=1}^{N} \frac{(x_i - \mu)^3}{\sigma^3}, \tag{2.8}$$

[4]For a review of the empirical evidence see Svetlozar T. Rachev, Christian Menn, and Frank J. Fabozzi, Fat-Tailed and Skewed Asset Return Distributions: Implications for Risk Management, Portfolio Selection [19].

where μ is the mean, σ the standard deviation, and N the number of data points $(x_i, \ldots, x_N)$.

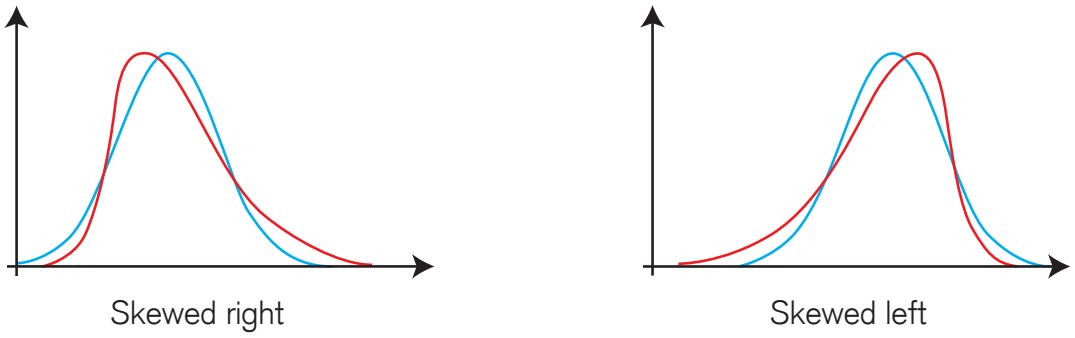

Figure 2.3: Skewness is a measure of symmetry or asymmetry of a distribution. The skewed distributions (red curves) are compared to the normal distributions (blue curves). Source: Credit Suisse [24].

The kurtosis expresses the peakedness of a distribution curve and indicates whether the likelihood of extreme events is greater than a normal distribution would suggest [6]. It is known as the fourth empirical moment. The more peaked the curve (leptokurtic), the greater the likelihood of extreme events. The flatter the curve (platykurtic), the less likely extreme events are to occur (see Figure 2.4). A normal distribution has a kurtosis of 3. The kurtosis is defined as

$$ku = \frac{1}{N-1} \sum_{i=1}^{N} \frac{(x_i - \mu)^4}{\sigma^4}, \tag{2.9}$$

where μ is the mean, σ the standard deviation, and N the number of data points $(x_i, \ldots, x_N)$.

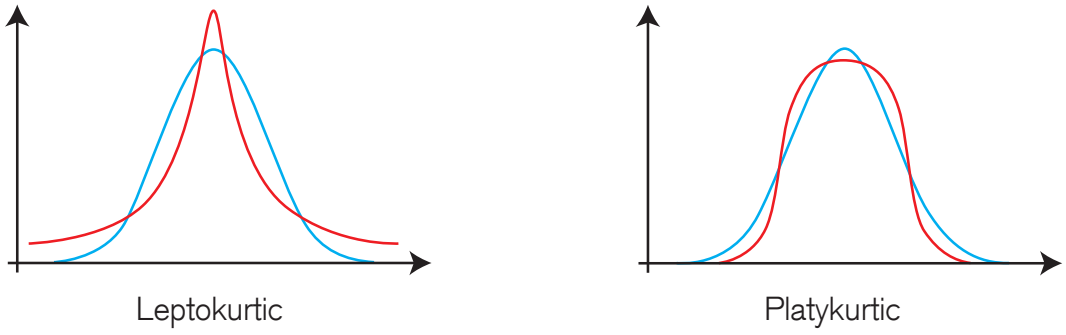

Figure 2.4: Investors prefer low kurtosis (platykurtic) distributions. The distributions with a kurtosis different from 0 (red curves) are compared to the normal distributions (blue curves). Source: Credit Suisse [24].

Distributions with a kurtosis greater than 3 are known as fat tails (see Figure 2.5). Investors prefer lower kurtoses to higher kurtoses – they do not like the higher probability of losses that happens in distributions with fat tails. An example of a skewed, leptokurtic distribution can be found, in the equity market returns, as demonstrated in Figure 2.6.

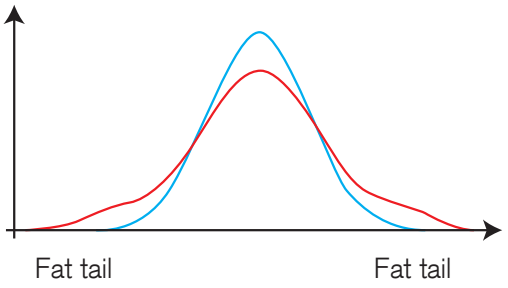

Figure 2.5: Fat tails refer to distributions (red curve) with a relatively high risk of extreme losses and profits; the extremes are more pronounced than in a normal distribution (blue curve). Source: Credit Suisse [24].

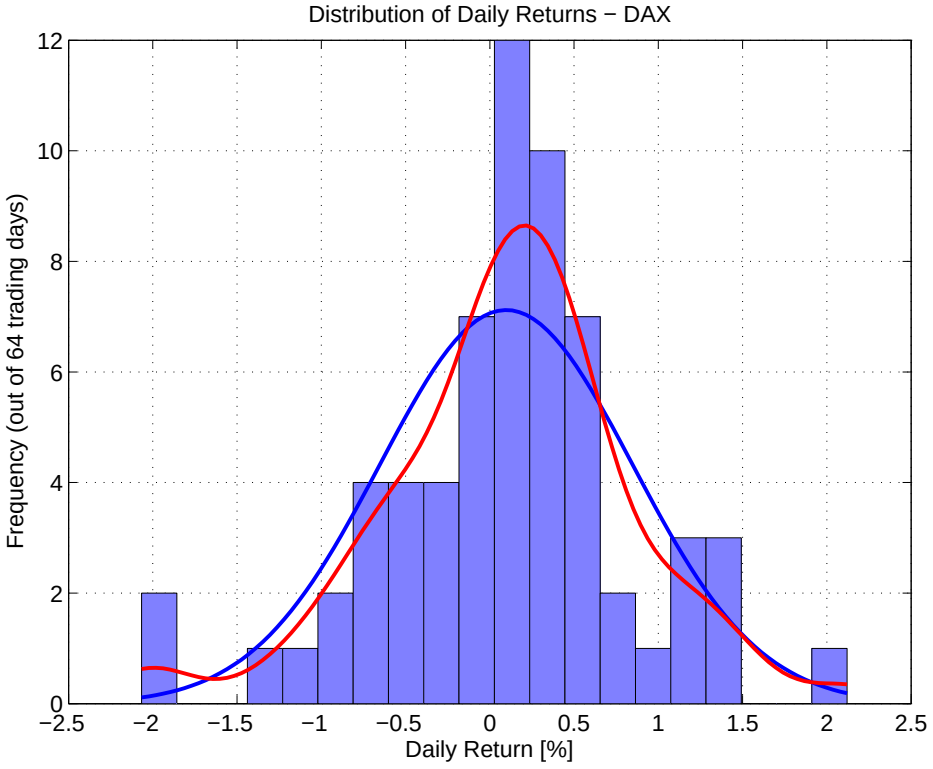

Figure 2.6: An example of an asymmetric distribution (red curve) that deviates from the normal distribution (blue curve) are the historical equity market daily returns of the DAX from October 3 to December 30, 2005. The probability distribution of the red curve is plotted as kernel smoothing approximation of the histogram, using the `ksdensity` function in Matlab.

2.3 Risk

Trading in assets whose future outcomes are uncertain necessarily involves risk for the investors. Thus, the management of risk is a major concern for the operation of financial markets [9]. For example:

- Financial regulators try to minimize the occurrence and impact of the collapse of financial institutions by placing restrictions on the types and sizes of permitted trades, such as limits on short sales.
- Risk managers in investment companies place restrictions on the activities of individual traders, seeking to avoid levels of exposure that the company may not be able to meet in extreme circumstances.
- Individual investors seek to diversify their investments to avoid extreme exposure to sudden moves in financial markets.

The mathematical analysis of risk measures has also been a principal concern of actuarial and insurance professions from the beginning. Also, it plays a fundamental role in the theory of portfolio selection where the objective is to find a portfolio that maximizes expected return while minimizing risk [9]. The following sections contain a description of which properties a good risk measure should have and an overview of volatility, the basis of the mean-variance portfolio optimization. In addition, VaR and CVaR are introduced as downside risk measures, which are further used throughout the thesis for the optimization approach in Chapter 3.

2.3.1 Coherent Risk Measures

Artzner et al. [2, 3] introduced a framework in which an ideal risk measure fulfills all of the framework's four axioms. These axioms define the minimum standard for any risk measure to be coherent. Even if a risk measure fails in one of the four axioms, it can lead to incorrect results and a improper estimation of risk. Jorion [14] defines them as following:

Axiom 1: Subadditivity
Merging assets from portfolios P_1 and portfolios P_2 cannot increase risk. The concept of diversification must have a reduction in risk ρ.

$$\rho(P_1 + P_2) \leq \rho(P_1) + \rho(P_2) \tag{2.10}$$

Axiom 2: Monotonicity
If portfolio P_1 has systemically lower returns than portfolio P_2, its risk must be greater.

$$if \quad P_1 \leq P_2, \quad then \quad \rho(P_1) \geq \rho(P_2) \tag{2.11}$$

Axiom 3: Translation invariance
The addition of cash in the amount k to a portfolio should reduce its risk by k.

$$\rho(P + k) = \rho(P) - k \tag{2.12}$$

Axiom 4: Homogeneity
Increasing the size of a portfolio by a factor b should simply scale its risk by the same factor.

$$\rho(bP) = b\rho(P) \tag{2.13}$$

2.3.2 Volatility

According to Elliott and Kopp [9], volatility is the simplest risk measure. It is the basis for the traditional portfolio theory and, therefore, used in Markowitz' mean-variance [12] and related portfolio optimization approaches. Volatility is a measure for variation in prices of an asset or a portfolio. The higher the volatility is, the greater are the price differences in a specific time period. Volatility σ estimated by a sampled mean is biased with

$$\sigma = \sqrt{\frac{1}{N-1}\sum_{i=1}^{N}(R_i - \mu)^2}, \tag{2.14}$$

where R_i represents the returns, N the number of returns, and μ the expected return (see Equation 2.3) [4].

The advantage of volatility as a risk measure is that it is very easy to calculate. However, even though the risk measure fulfills the axioms of the Artzner's framework [2], there are some disadvantages to it. The most important disadvantage is that the variance measure treats returns and losses symmetrically. Volatility is, therefore, solely ideal for normal distributed returns. The recent past – especially when the markets crashed in 2001 and in 2007 – has shown that returns do not necessarily follow the logic of a normal probability distribution [12]. In fact, extreme losses occur more frequently than presumed by a normal distribution, which makes volatility less favorable to measure risk, because investors attach greater importance to

losses than to profits of the same magnitude. Nowadays, greater emphasis is placed on estimating the risk of losses [12]. By using downside risk measures, such as Value-at-Risk and Conditional Value-at-Risk, the risk of so called fat-tail distributions can be expressed and controlled more precisely.

2.3.3 Value-at-Risk

According to Jorion [14], Value-at-Risk is the worst loss (which is the maximum negative return) over a certain period of time which will not be exceeded with a given level of confidence β (see Figure 2.7). VaR has become the standard benchmark for measuring financial risks [9]. It is defined by the smallest number α at the confidence level $\beta \in (0, 1)$ such that the probability that the loss L exceeds α is at maximum (1 - β).

$$VaR_\beta = min\{\alpha \in \mathbb{R} : P(L > \alpha) \leq 1 - \beta\} \tag{2.15}$$

VaR was introduced by JP Morgan in 1994. It was initially considered to represent a new risk measure which doesn't lack on the issues of variance as a measure of risk (see Section 2.3.2). But after a while concern was raised because VaR has some unfavorable properties to be used as risk measure. For non-normal distributions VaR

- is not a coherent risk measure because it does not fulfill the subadditivity axiom, which implies that a portfolio's VaR might be higher than the sum of its assets (see Section 2.3.1),
- is non-linear,
- is non-convex and non-smooth with respect to portfolio positions, and
- has multiple local extrema.

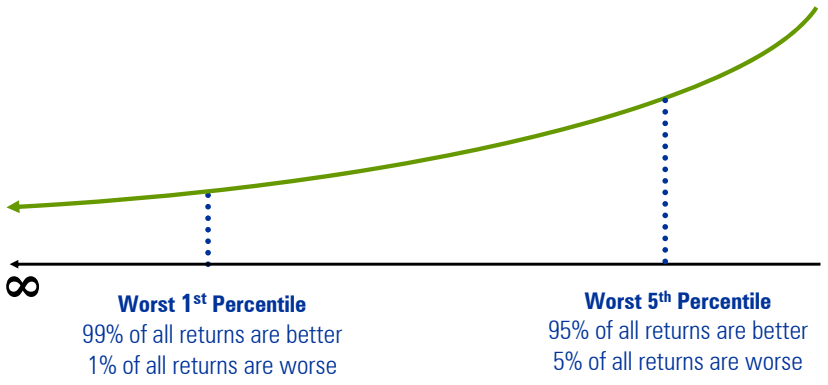

Figure 2.7: VaR identifies the return at a specific point in a return distribution (e.g., 1st or 5th percentile). Source: Ibbotson [5].

This means that VaR is difficult to optimize for discrete distributions, when it is calculated using scenarios. Due to its non-linearity it cannot be used for optimization models based on linear solvers. In the debate between Aaron Brown and David Einhorn, Einhorn compared VaR to "an airbag that works all the time, except when you have a car accident." [8]

2.3.4 Conditional Value-at-Risk

Like VaR, Conditional Value-at-Risk is a downside risk measure, but as shown in Figure 2.8, CVaR measures the average loss in the *entire tail* with a certain level of probability (e.g., 90%, 95%, 99%). Acerbi and Tasche [1] defined expected shortfall similarly to CVaR.

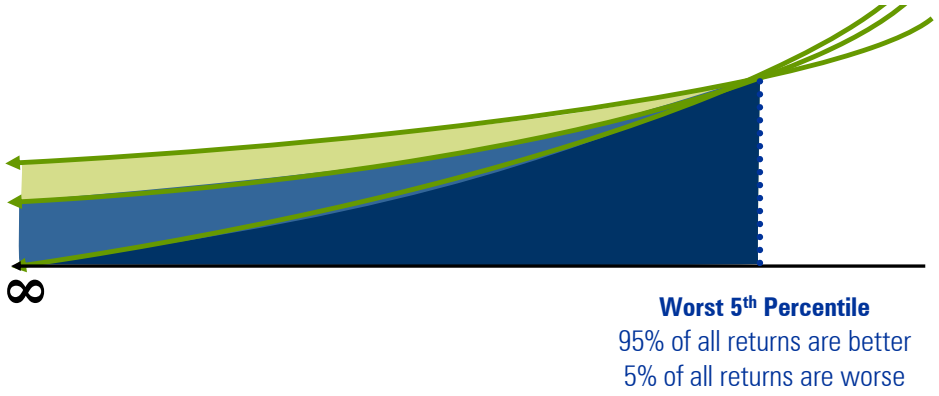

Figure 2.8: Different return distributions can have the same VaR's, but different CVaR's because of different tails. Source: Ibbotson [5].

CVaR is the average return given that the return is smaller than VaR with a certain level of confidence (see Figure 2.9). Rockafellar and Uryasev [20] defined CVaR for a specified confidence level β as

$$CVaR_\beta = \frac{1}{1-\beta} \int_{-\inf}^{VaR_\beta} f(w,r)p(r)\mathrm{d}y, \tag{2.16}$$

where $f(w,y)$ is the return function with distribution $p(r)$ of portfolio returns r over a certain time period. VaR_β is calculated over the same time period with the confidence level β.

Sarykalin et al. [22] considered the measure to be more consistent than VaR because CVaR

- is a coherent risk measure in the sense of Artzner et al. [2] (see Section 2.3.1),
- is applicable to non-symmetric loss distributions,
- can be implemented with linear programming (fast and stable) which allows the optimization of very large problems (over 1'000'000 instruments and scenarios),
- is convex and smooth with respect to portfolio positions, and
- accounts for risks beyond VaR, which means it is more conservative than VaR.

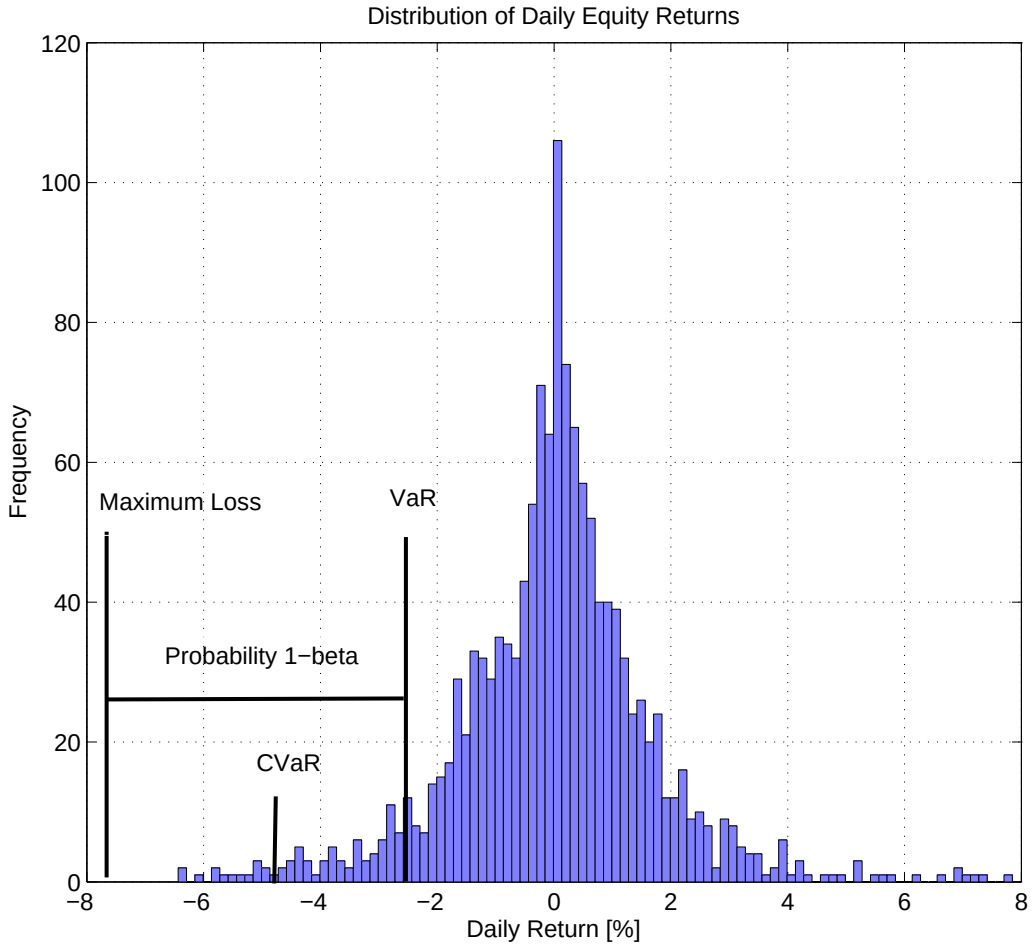

Figure 2.9: CVaR is the average return given that the return is smaller than VaR with a certain level of confidence β.

2.4 Portfolio Optimization

2.4.1 Diversification as Risk Reduction

Regardless of the optimization method to allocate assets in portfolios, the concept of diversification has established as very effective to reduce risk. However, the effect diminishes with increasing number of assets in the portfolio. Typically, the diversification effect is most efficient in portfolios composed of about 20 assets [10]. This means that diversification reduces risk only to a certain degree.

2.4.2 Efficient Portfolios and the Optimal Portfolio

Drake and Fabozzi [7] stated that portfolios are efficient when they provide the maximum possible expected return for a certain risk. To build efficient portfolios one needs to define some assumptions about investors and their behavior. The first assumption is that investors are risk-averse, which means that they will choose the portfolio with the lowest risk, when faced with several portfolios with the same expected return, but with different risk. On the other hand, a risk-averse investor will choose the portfolio with the highest return, when they have to choose from a set of portfolios with the same risk, but different expected returns. That means that efficient portfolios are located on the efficient frontier as shown in Figure 2.10. From a set of efficient portfolios, the optimal portfolio is the one that is most preferred by an investor.

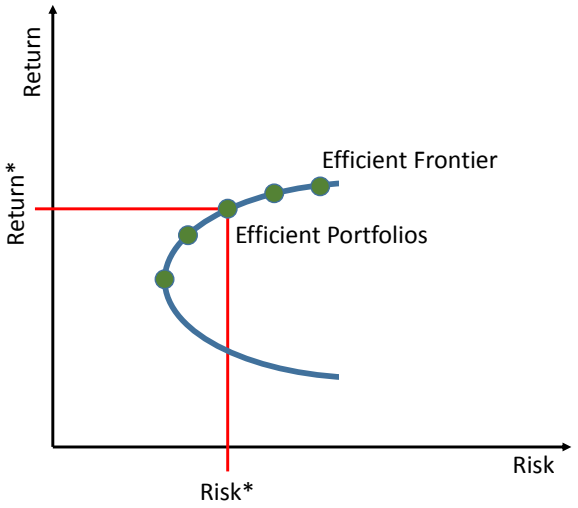

Figure 2.10: Efficient portfolios (green dots) with corresponding *Risk** and *Return** are on the efficient frontier.

2.4.3 Mean-Variance Optimization

Mean-variance is the traditional optimization approach introduced by Markowitz in his seminal paper titled "Portfolio Selection", published in 1952 [16]. Even before Markowitz presented his approach, portfolio theory was an important topic. However, the main focus of Bachelier[5] and his successors was to improve performance. Markowitz focused on risk. He established volatility (see Section 2.3.2) as the major risk measure in portfolio theory and showed how the risk can be reduced by diversification. He demonstrated how to generate financial portfolios, which have a maximum expected return for a given level of risk, measured in standard deviation σ. The portfolio with the lowest risk is called the minimum variance portfolio (see Figure 2.11).

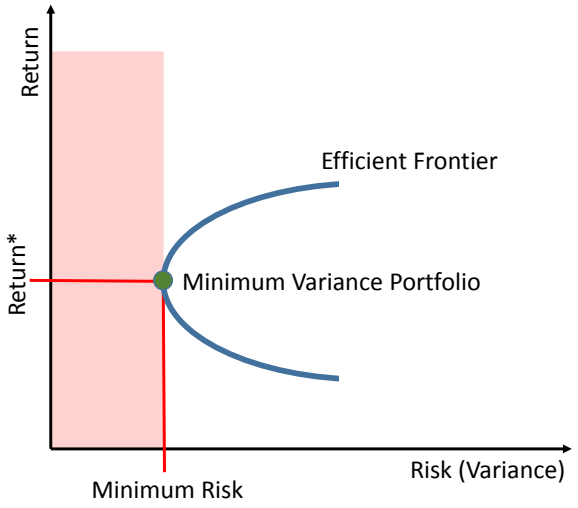

Figure 2.11: The *Minimum Variance Portfolio* (green dot) on the efficient frontier has the *Minimum Risk* with corresponding *Return**.

The reason for the preference of the mean-variance model is the fact that expected portfolio returns are difficult to estimate and that estimation errors can lead to sub-optimal portfolio selection [13]. When taking into account that all assets in the portfolio have the same expected return, they only differ with respect to their risk. With this expected return assumption, and the assumption that variance is an appropriate measure of risk, the mean-variance model is a very powerful method because the model depends only on the portfolio covariance matrix, which can be estimated precisely. In addition, the model has the advantage that it is easy to understand and straightforward to use. However, because volatility, as the basis of the mean-variance optimization, has known limitations (see Section 2.3.2), the development of new optimization approaches continues.

[5] Louis Bachelier (1870–1946) is recognized as the founder of financial mathematics [24]. He used Brownian motion to model price trends of securities as a symmetrical fluctuation away from the expected returns.

Chapter 3

Portfolio Optimization with CVaR Constraints

Because mean-variance and other optimization frameworks based on symmetric distributions penalize positive as well as negative returns, alternative risk measures, for example VaR (see Section 2.3.3), have become more important in recent years. However, VaR is neither an ideal basis for portfolio optimization due to the issues described in Chapter 2. CVaR, in contrast to VaR, captures tail risk more precisely and is accepted as a coherent risk measure, applicable for non-normal distributions and has the ability to be implemented with linear programming (see Section 2.3.4). In this chapter, the optimization approach of Rockafellar and Uryasev [20] to optimize portfolios with CVaR is elaborated and implemented.

3.1 Optimization Approach

The approach of Rockafellar and Uryasev [20] is considered to optimize portfolios of assets either by a minimized CVaR, or by putting limits on the CVaR under which the portfolio can be optimized.

3.1.1 Derivation

We use the notation that $f(w, r)$ is the loss function associated with the decision vector $w \in W \subset \mathbb{R}^n$ (e.g. portfolio weights) and the random vector $r \in \mathbb{R}^m$ (e.g. returns of portfolio assets). When r has the distribution $p(r)$, then the probability of the loss $f(w, r)$ not exceeding a certain threshold α is given by

$$\Psi(w,\alpha) = \int\limits_{f(w,r)\leq\alpha} p(r)\mathrm{d}r, \tag{3.1}$$

where Ψ, as a function of α for fixed w, is the cumulative distribution function for the loss related with w. $\Psi(w,\alpha)$ is non-decreasing with respect to α and, as with $p(r)$, is also continuous. The VaR and CVaR values can be denoted as $\alpha_\beta(w)$ and $\phi_\beta(w)$ respectively, in the description of Rockafellar and Uryasev [20] they are given by

$$\alpha_\beta(w) = min\{\alpha \in \mathbb{R} : \Psi(w,\alpha) \geq \beta\} \tag{3.2}$$

and

$$\phi_\beta(w) = (1-\beta)^{-1} \int\limits_{f(w,r)\geq\alpha_\beta(w)} f(w,r)p(r)\mathrm{d}r. \tag{3.3}$$

This means that CVaR is the integral of the losses $f(w,r)$ which are greater or equal than the VaR ($\alpha_\beta(w)$) divided by $(1-\beta)$ where β is the confidence level (e.g., 95%; see Figure 2.9).

3.1.2 Reformulation

The CVaR function in Equation 3.3 is difficult to handle as it is a function including the VaR-function 3.2. Using $\phi_\beta(w)$ for the optimization of CVaR implies that VaR would have to be calculated first. The major contribution by Rockafellar and Uryasev [20] was the derivation of a CVaR-function that is independent of the VaR-function, making the optimization process less complicated. This function is defined by

$$\begin{aligned} F(w,\alpha) &= \alpha + (1-\beta)^{-1} \int\limits_{r\in\mathbb{R}^m} [f(w,r)-\alpha]^+ p(r)\mathrm{d}r, \\ &\text{where} \\ [t]^+ &= \begin{cases} t & when \quad t > 0, \\ 0 & when \quad t \leq 0. \end{cases} \end{aligned} \tag{3.4}$$

Rockafellar and Uryasev [20] show that minimizing the function $F(w,\alpha)$ gives the same result as solving $\phi_\beta(w)$.

$$\phi_\beta(w) = \min_{\alpha\in\mathbb{R}} F(w,\alpha) \tag{3.5}$$

The crucial feature of $F(w, \alpha)$, in Equation 3.4, is that not just portfolio weights w, but also the quantile level α must be optimized. This means that by implementing the CVaR optimization approach VaR is calculated as side-effect. Rockafellar and Uryasev [20] prove that usually the minimization of CVaR also leads to a nearly optimal VaR because VaR never exceeds CVaR. Therefore, portfolios with low CVaR must have low VaR as well. Additionally, it is a convex function, which is a key property for optimization[6].

3.1.3 Discretization

The integral of Equation 3.4 is approximated by sampling the probability distribution of r according to its density $p(r)$. If the sampling generates a collection of values $(r_1, \ldots, r_J)$, the approximation has the form

$$F(w, \alpha) = \alpha + (1 - \beta)^{-1} \sum_{j=1}^{J} \pi_j [f(w, r) - \alpha]^+, \tag{3.6}$$

where J is the number of scenarios and π_j are probabilities of the scenarios r_j. To illustrate the approach of Rockafellar and Uryasev [20] as a portfolio optimization method, we consider that the decision vector w represents a portfolio of financial assets in the sense that $w = (w_1, \ldots, w_N)$ where N is the number of assets in the portfolio and w_i is the weight of the asset at the position i in the portfolio so that

$$w_i \geq 0 \quad for \quad i = 1, \ldots, N \quad with \quad \sum_{i=1}^{N} w_i = 1. \tag{3.7}$$

The return on a portfolio is the sum of the returns on the individual assets in the portfolio, scaled by the asset weights w_i. Because the derivation of CVaR was elaborated with the loss we take the negative of the returns, given by

$$f(w, r) = -[w_1 r_1 + \ldots + w_N y_N] = -w^T r. \tag{3.8}$$

By using simulation methods, which will be described in Chapter 4 to generate returns, the CVaR minimization problem is an optimization of the arguments w (asset weights in the portfolio) and α (VaR).

[6] Convexity eliminates the possibility of a local minimum being different from a global minimum.

$$\begin{aligned} \underset{(w,\alpha)}{\operatorname{argmin}} \quad & \alpha + \frac{1}{J(1-\beta)} \sum_{j=1}^{J} [-w^T r_j - \alpha]^+ \\ s.t. \quad & w \in W, \alpha \in \mathbb{R} \end{aligned} \tag{3.9}$$

3.1.4 Linearization

The minimization of the objective function in Equation 3.9 is not a linear program because the expression $[-w^T r_j - \alpha]^+$, which ensures that only losses are taken into account which are greater than or equal to VaR, has to be implemented with a non-linear operator (e.g., `max`). The objective function could be implemented in Matlab as:

```
i = 1:nAssets;
objfun = @(w) w(nAssets+1)+(1/nSims)*(1/(1-beta))*sum(max(-w(i)*r(:,i)'-w(nAssets+1),0));
```

For solving the optimization problem by a linear solver, it has to be re-written with the introduction of auxiliary variables u_j, $j = 1, \ldots, J$ to the form

$$\begin{aligned} \underset{(w,\alpha,u_1,\ldots,u_J)}{\operatorname{argmin}} \quad & \alpha + \frac{1}{J(1-\beta)} \sum_{j=1}^{J} u_j \\ s.t. \quad & w \in W, \alpha \in \mathbb{R} \\ & u_j \geq 0, \quad j = 1, \ldots, J \\ & w^T r_j + \alpha + u_j \geq 0, \quad j = 1, \ldots, J \end{aligned} \tag{3.10}$$

The solution is that the non-linear `max`-function, which has been used in the non-linear problem in Equation 3.9, is replaced by the variables u_j and additional linear constraints on these variables. Assume that $-w^T r_j - \alpha$ is negative[7], which means $[-w^T r_j - \alpha]^+ = 0$, then $w^T r_j + \alpha$ is positive and u_j can be zero because the constraint is already fulfilled. For the other scenario where $-w^T r_j - \alpha$ is positive[8], which means $[-w^T r_j - \alpha]^+ = -w^T r_j - \alpha$, then u_j cannot be less than $-w^T r_j - \alpha$ in order for the constraint $w^T r_j + \alpha + u_j \geq 0$ in Equation 3.10 to be fulfilled.

By applying this function as the objective function in a linear solver, the portfolio allocation with the lowest risk will be found. On the efficient frontier in Figure 3.1 this is the *Optimal Portfolio* which has a return of $Return^*$ and the lowest possible risk (*Minimum Risk*).

[7] the loss is smaller than VaR
[8] the loss is greater than or equal to VaR

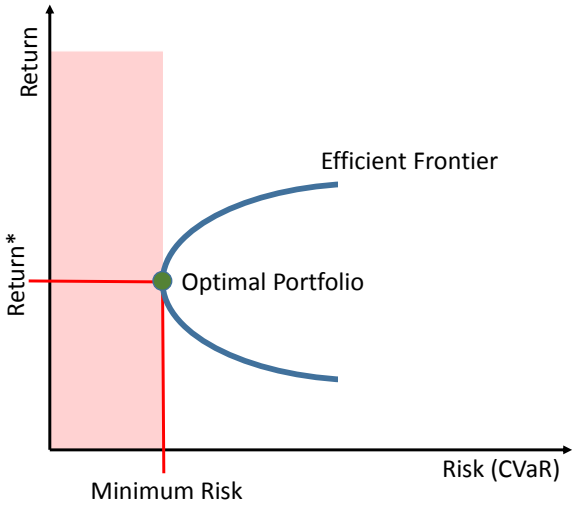

Figure 3.1: The *Optimal Portfolio* in a minimal CVaR sense on the efficient frontier with *Minimum Risk* and *Return**.

3.2 Linear Programming

Due to the linearization in the Rockafellar and Uryasev [20] approach described in Section 3.1.4, the optimization problem can now be solved using linear programming.

3.2.1 Description

Linear programming (LP) is a method for the optimization of linear objective functions, subject to linear equality and linear inequality constraints. The intersections of the linear constraints define a feasible region in the form of a convex polyhedron (see Section 3.2). A linear programming algorithm finds a point in the polyhedron where the objective function has the smallest (or largest) value, if such a point exists [27]. For a minimization problem it has the form

$$\begin{aligned} \min \quad & f^T w \\ s.t. \quad & Aw \leq b \\ and \quad & x \geq 0, \end{aligned} \tag{3.11}$$

where w represents the vector of variables which have to be optimized. f and b are vectors of known coefficients, A is a known matrix of coefficients. The inequalities $Aw \leq b$ and $x \geq 0$ are the constraints which specify a convex polyhedron (feasible region) over which the objective

function is to be optimized. As an example for a simple linear program the objective function (the coefficient vector of the objective function respectively)

$$f^T w = -3w_1 - 2w_2 = \begin{matrix} w_1 & w_2 \end{matrix} \begin{pmatrix} -3 & -2 \end{pmatrix} \tag{3.12}$$

is subjected to the linear inequality constraints in the form $A * w \leq b$

$$A = \begin{matrix} w_1 & w_2 \end{matrix} \begin{pmatrix} 3 & 4 \\ 2 & 1 \end{pmatrix} \qquad b = \begin{pmatrix} 7 \\ 3 \end{pmatrix} \tag{3.13}$$

and the linear equality constraints in the form $Aeq * w = beq$

$$Aeq = \begin{matrix} w_1 & w_2 \end{matrix} \begin{pmatrix} -3 & 2 \end{pmatrix} \qquad beq = \begin{pmatrix} 2 \end{pmatrix} \tag{3.14}$$

which results in a polyhedron, defined by the intersections of the linear constraints, illustrated in Figure 3.2.

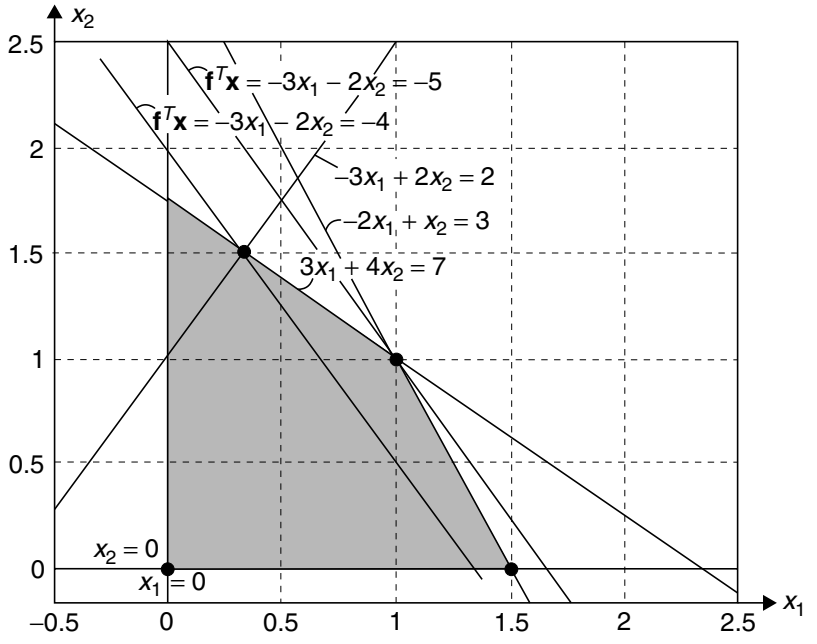

Figure 3.2: A convex polyhedron, formed by intersections of linear constraints, defines the feasible region for solutions of the objective function. Source: Applied Numerical Methods Using Matlab [27].

3.2.2 Advantages

When a problem can be implemented with linear programming:

- optimal solutions are always found at 'corner points' of the constraint polyhedron (where two or more constraints intersect). This means that a LP solver needs to consider many fewer points than a non-linear programming solver which implies short solving times.
- it can be used to solve very large problems with millions of variables
- it is always possible to determine that an linear programming problem has:
  - no feasible solution,
  - an unbounded objective, or a globally optimal solution

3.2.3 Solvers

Available LP solvers differ in many ways. They come with different licenses and of course different features, for example in terms of how problems can be specified. Popular and well-known commercial LP solvers [17] (license cost of approx. 7'000 - 10'000 USD) are:

- CPLEX: is now actively developed by IBM. The software also features several interfaces that make it possible to connect the solver to different program languages and programs. However, also a stand-alone executable is provided.
- XPRESS: is available on most common computer platforms and also provides several interfaces including a callable library APIs for several programming languages as well as a standalone command-line interface.

3.2.4 Advantage for the Optimization Approach

Because the CVaR optimization approach of Rockafellar and Uryasev can be solved with a linear solver, it is possible to optimize:

- very large portfolios,
- and a large number of scenarios,
- with normal as well as non-normal distributions of the underlying asset returns, and
- with relatively small computational resources [26].

This is a major advantage over the traditional mean-variance optimization of Markowitz which is only optimal for normal distributions and can only be solved with quadratic programming, which uses a lot more computational resources than LP.

3.3 Maximizing Return for a Certain Risk

Instead of optimizing the portfolio to have the lowest possible risk (described in Section 3.1.4), a more useful solution is to find the portfolio allocation with the highest return for a certain risk level [15]. This means that we want to maximize the expected return (which is minimizing the expected loss) for a certain CVaR-budget.

3.3.1 Problem Definition

The linearized optimization problem of Equation 3.10 has to be re-formulated to a new optimization problem of the form

$$\begin{aligned} \underset{(w,\alpha,u_1,\ldots,u_J)}{\operatorname{argmin}} \quad & -w^T\bar{m} \\ s.t. \quad & w \in W, \alpha \in \mathbb{R} \\ & \alpha + \frac{1}{J(1-\beta)} \sum_{j=1}^{J} u_j \leq \delta \\ & u_j \geq 0, \quad j = 1,\ldots,J \\ & w^T r_j + \alpha + u_j \geq 0, \quad j = 1,\ldots,J \end{aligned} \tag{3.15}$$

where δ is the CVaR limit and $\bar{m}$ is the expected outcome of r (in the context of the current thesis, $\bar{m}$ is the expected return $E(R)$). This means that the CVaR function of Rockafellar and Uryasev [20] is a constraint of the objective function. As shown in Figure 3.3, the optimization of Equation 3.15 finds the portfolio allocation with the maximum return ($Return^*$) on the efficient frontier within a certain risk limit, expressed with CVaR. This is the optimal portfolio on the efficient frontier for a given risk in a CVaR sense. Increasing the risk limit will result in a higher expected return.

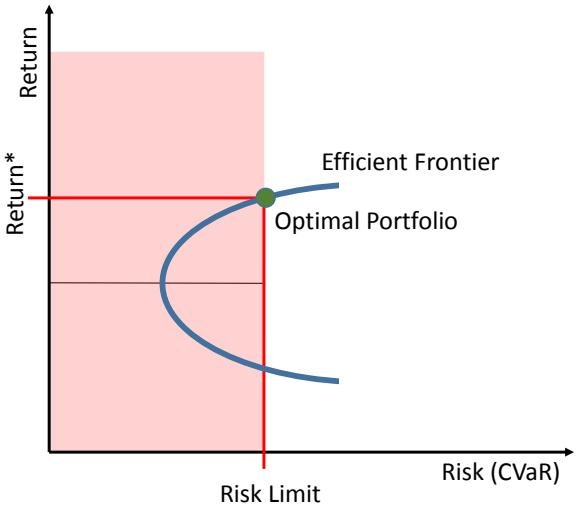

Figure 3.3: The *Optimal Portfolio* on the efficient frontier with maximum return *Return** for a certain *Risk Limit*, expressed with CVaR.

3.3.2 Implementation

The optimization of Equation 3.15 is implemented with the Matlab function `fmincon`[9]. The initialization of the parameters for an implementation with an LP solver (e.g. `linprog`) is similar to `fmincon`. It returns the vector `w`, where $w_1, \ldots, w_N$ are the asset weights for the optimal portfolio and w_{N+1} is the corresponding VaR. The return value `fval` is the expected return under the corresponding constraints. The parameters for `fmincon` will be elaborated step by step[10]

```
[w, fval] = fmincon(objfun, w0, A, b, Aeq, beq, LB, UB, [], options);
```

3.3.2.1 Objective Function

As can be seen in the `fmincon` function call above, an objective function has to be defined and passed as first parameter. The aim is to maximize the return, which is the same as to minimize the loss $-w^T m$. In this case the objective function has the form:

```
objfun = @(w) -mean(r(:,1:nAssets))*w(1:nAssets)';
```

[9] fmincon is a general constrained optimization routine. It finds the minimum of a constrained multivariable function.

[10] the `options` parameter is not part of the concept and is, therefore, not explained in this section. The full source code of the implementation can be found in Appendix A.

3.3.2.2 Inequality Constraints

The linear inequality constraints have to be in the form $A * w \leq b$. The vector w contains the variables for the asset weights $(w_1, \ldots, w_N)$, VaR (α) and the auxiliary variables $(u_1, \ldots, u_J)$ which were introduced in Equation 3.10. The first row in A and b represents $\alpha + \frac{1}{J(1-\beta)} \sum_{j=1}^{J} u_j \leq \delta$. The remaining rows substitute $w^T r_j + \alpha + u_j \geq 0, j = 1, \ldots, J$. The vector $r_{11}, \ldots, r_{JN}$ represents the returns of the corresponding assets in the portfolio $(1, \ldots, N)$ associated with the simulations $(1, \ldots, J)$. Because the objective is to minimize losses, the returns have to be multiplied with -1. This means that A is a $N + 1 + J$ dimensional matrix, where N is the number of assets in the portfolio and J is the number of simulations. The additional column is used to represent VaR (α). So the matrices are defined as follows:

$$A = \begin{array}{c} \begin{matrix} w_1 & w_2 & \cdots & w_N & \alpha & u_1 & u_2 & \cdots & u_J \end{matrix} \\ \begin{pmatrix} 0 & 0 & \cdots & 0 & 1 & \frac{1}{J*(1-\beta)} & \frac{1}{J*(1-\beta)} & \cdots & \frac{1}{J*(1-\beta)} \\ -r_{11} & -r_{12} & \cdots & -r_{1N} & -1 & -1 & 0 & \cdots & 0 \\ -r_{21} & -r_{22} & \cdots & -r_{2N} & -1 & 0 & -1 & \cdots & 0 \\ \vdots & \vdots & \ddots & \vdots & \vdots & \vdots & \vdots & \ddots & \vdots \\ -r_{J1} & -r_{J2} & \cdots & -r_{JN} & -1 & 0 & 0 & \cdots & -1 \end{pmatrix} \end{array} \quad b = \begin{pmatrix} -\delta \\ 0 \\ 0 \\ 0 \\ 0 \end{pmatrix} \tag{3.16}$$

3.3.2.3 Lower and Upper Bounds

The lower bound (LB) and upper bound (UB) vectors are used to define:

- a maximum asset weight: $UB_1, \ldots, UB_N$ for $w_1, \ldots, w_N$
- a minimum asset weight: $LB_1, \ldots, LB_N$ for $w_1, \ldots, w_N$
- the constraint $u_j \geq 0, \quad j = 1, \ldots, J$: $u_1 \ldots, u_J = 0$ in LB

Thus, they have the form:

$$UB = \begin{array}{c} \begin{matrix} w_1 & w_2 & \ldots & w_N & \alpha & u_1 & u_2 & \ldots & u_J \end{matrix} \\ \begin{pmatrix} UB_1 & UB_2 & \ldots & UB_N & \text{inf} & \text{inf} & \text{inf} & \ldots & \text{inf} \end{pmatrix} \end{array} \tag{3.17}$$

$$LB = \begin{array}{c} \begin{matrix} w_1 & w_2 & \ldots & w_N & \alpha & u_1 & u_2 & \ldots & u_J \end{matrix} \\ \begin{pmatrix} LB_1 & LB_2 & \ldots & LB_N & 0 & 0 & 0 & \ldots & 0 \end{pmatrix} \end{array} \tag{3.18}$$

An upper bound of 1 for an asset position means that each asset can have a maximum of 100% of the portfolio weight. Defining an upper bound smaller than 1 is used if there are policies not allocating more than a certain level for a single asset (e.g., 50%) to specify a maximum concentration. Similarly, a lower bound is used if there are policies allocating assets with a certain minimum of the portfolio weight (e.g., 5%). The lower bound as well as the upper bound vector can contain different values for the several asset positions so that different minimum or maximum weight levels can be specified (e.g., 5%, 10%, 30%, ...).

3.3.2.4 Equality Constraints

Similar to the linear inequality constraints, the linear equality constraints have to be in the form $Aeq * w = beq$. In this context the equality matrices are used to define

$$\sum_{i=1}^{N} w_i = 1 \quad for \quad i, \ldots, N \tag{3.19}$$

which means that the sum of the assets $(w_1, \ldots, w_N)$ is 100%, and therefore, the matrices for the inequality constraints have the form

$$Aeq = \begin{array}{c} \begin{array}{ccccccccc} w_1 & w_2 & \ldots & w_N & \alpha & u_1 & u_2 & \ldots & u_J \end{array} \\ \begin{pmatrix} 1 & 1 & \ldots & 1 & 0 & 0 & 0 & \ldots & 0 \end{pmatrix} \end{array} \qquad beq = \begin{pmatrix} 1 \end{pmatrix} \tag{3.20}$$

3.3.2.5 Initial Values

The last parameter for `fmincon` is the vector $w0$ which is used by the optimizer for the initial values of the several variables. The initial values for the variables $u_1, \ldots, u_J$ have the value 0, the values for $w_1, \ldots, w_N$ are initialized with 1^{-N} and the initial value for α is the *quantile* of the equally weighted portfolio returns, named $VaR0$. The vector $w0$ has the form

$$w0 = \begin{array}{c} \begin{array}{ccccccccc} w_1 & w_2 & \ldots & w_N & \alpha & u_1 & u_2 & \ldots & u_J \end{array} \\ \begin{pmatrix} \frac{1}{N} & \frac{1}{N} & \ldots & \frac{1}{N} & VaR0 & 0 & 0 & \ldots & 0 \end{pmatrix} \end{array} \tag{3.21}$$

Chapter 4

Scenario Generation

The optimization approach of Rockafellar and Uryasev [20], described in Chapter 3, requires generated scenarios which simulate future prices of the financial assets underlying the portfolio. In this chapter two different methods of scenario generation are demonstrated. The first is a Monte Carlo simulation based on geometric Brownian motion, and the second is a bootstrapping of historical data.

4.1 Monte Carlo Simulation

For the Monte Carlo simulation we use geometric Brownian motion, a continuous-time stochastic process in which the logarithm of the randomly varying quantity follows a Brownian motion[11] with drift [21]. It is a stochastic processes satisfied by the solution to a stochastic differential equation (SDE), as demonstrated in Equation 4.1.

4.1.1 Asset Paths

The simulation of a potential future asset price S_t is based on the Wiener process of the form

$$S_t = S_0 \exp[(\mu - \frac{\sigma^2}{2})t + (\sigma\sqrt{t})\varepsilon] \tag{4.1}$$

where

- S_0: Asset price today
- S_t: Asset price in the future

[11] also called a Wiener process

- t: Time increment
- μ: Expected return (the percentage drift)
- σ: Expected volatility
- ε: Random number ($\varepsilon \sim N(0,1)$)

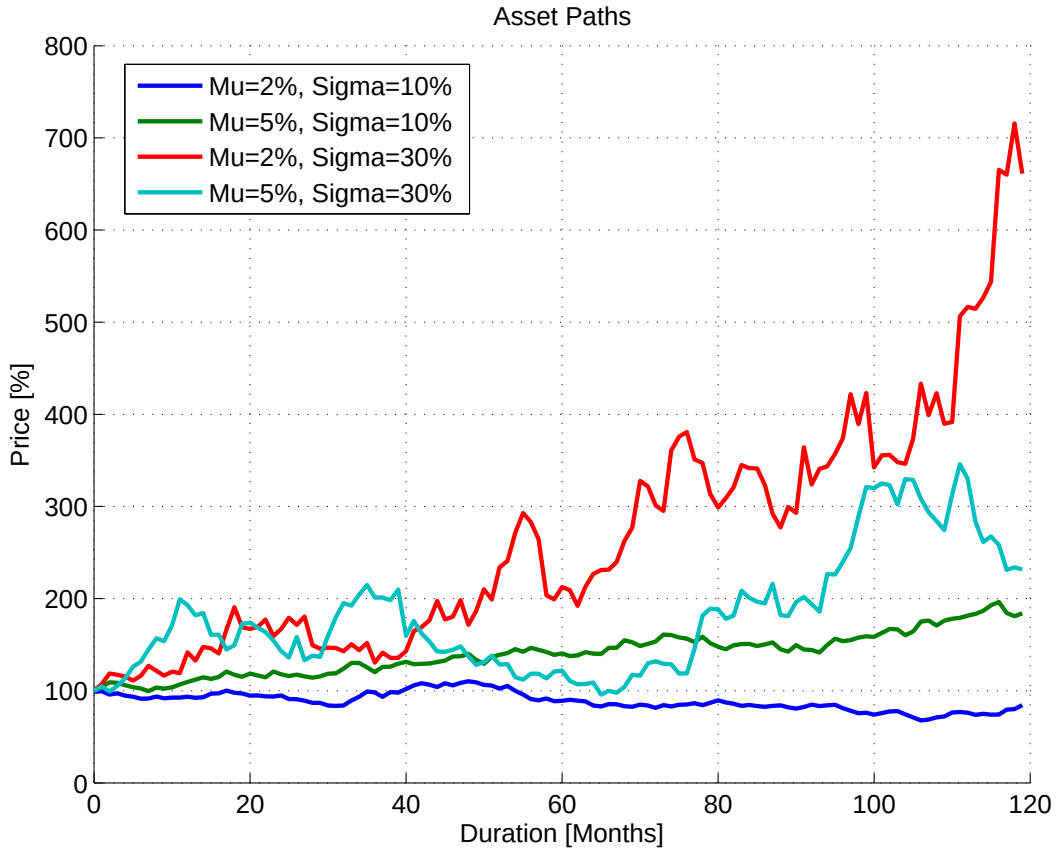

Figure 4.1: Example of a random path of four asset prices for 10 years, generated with geometric Brownian motion with different expected values μ and different standard deviations σ for each asset.

The repeated use of Equation 4.1 generates multiple potential future asset paths, as shown in Figure 4.1. For a meaningful Monte Carlo simulation many thousands of future asset paths need to be generated. By increasing the number of generated paths, the time for the simulation as well as the optimization increases. However, the more simulations are generated, the more likely the effects of tail events and outliers will be accounted for the optimization.

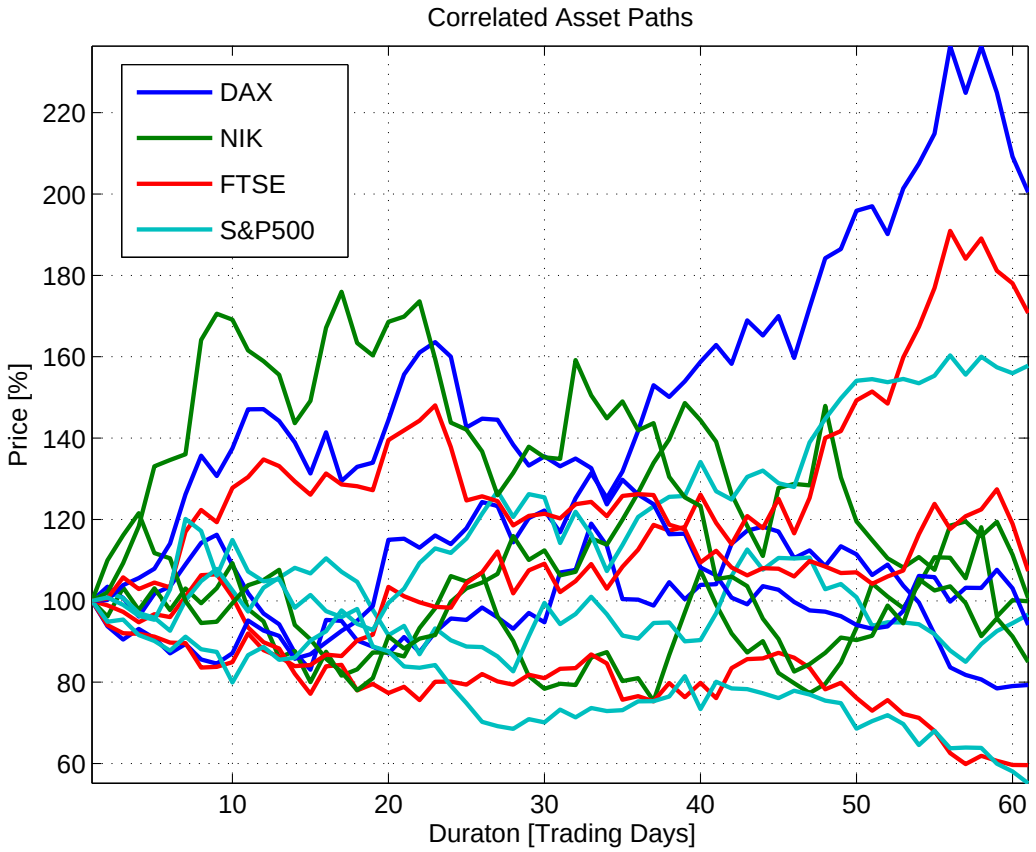

Figure 4.2: Example of three correlated asset price simulations for 60 trading days, generated with geometric Brownian motion. The expected return μ and standard deviation σ are from the underlying price indices (DAX, NIK, FTSE, S&P500) measured in the time period from October 3 to December 30, 2005.

4.1.2 Correlated Asset Paths

In order to have meaningful scenarios it is not enough to generate independent random paths for each asset. The generated paths need to correlate the assets they model, as shown in Figure 4.2. Nyholm [18] suggests to use the Cholesky decomposition of the covariance matrix to generate correlated asset paths. This decomposition allows for the calculation of the upper triangular matrix U from a positive definite matrix[12], in this case the covariance matrix C[13]. The Cholesky decomposition has the form (see e.g., Glasserman [11]):

$$C = U^T U \tag{4.2}$$

[12]The matrix $C_{[n,n]}$ is positive definite if $w^T C w > 0$, $\forall w$ where $w_{[n,1]} \neq 0_{[n,1]}$.
[13]A covariance matrix is always positive definite

The upper triangular matrix U is useful for generating correlated random numbers in the following way:

$$R_{r,c} = W_{r,c} * U_{c,c} \tag{4.3}$$

First, uncorrelated random numbers W have to be generated. Then, these random numbers are 'run through' the Cholesky decomposition to generate the matrix R of correlated random numbers of the desired dimension, here (r, c). The corresponding Matlab function to generate sample correlated paths for assets assuming geometric Brownian motion (see Equation 4.1) is:

```
function [S] = gbmcorr(S0, mu, sig, corr, dt, nSteps, nSims)
    nAssets = length(S0);
    drift = mu - sig.^2/2;
    U = chol(corr);
    S = nan(nSteps+1,nAssets,nSims);

    for idx = 1:nSims
        W = randn(nSteps, nAssets);
        R = W*U;
        S(:,:,idx) = [ones(1,nAssets); ...
          cumprod(exp(repmat(drift*dt,nSteps,1) ...
          +R*diag(sig)*sqrt(dt)))]*diag(S0);
    end
end
```

where `S0` is a vector with the initial-prices of the several assets, `mu` is the expected return, `corr` is the correlation coefficient matrix, `dt` is the increase in time, `nSteps` is the number of time steps T, and `nSims` is the number of simulations J.

The result is a T-by-N-by-J dimensional matrix with asset prices where each row represents a time step $(t_1, \ldots, t_T)$, each column represents a different asset $(a_1, \ldots, a_N)$, and each slice in the 3^{rd} dimension represents a separate simulation run $(S_1, \ldots, S_J)$. Equation 4.4 is an example of a correlated asset path output with three simulation runs and a period of four trading days:

$$S_1 = \begin{array}{c} \\ t_1 \\ t_2 \\ t_3 \\ t_4 \end{array} \begin{array}{c} \begin{array}{ccc} a_1 & \ldots & a_N \end{array} \\ \begin{pmatrix} p_{111} & \ldots & p_{1N1} \\ p_{211} & \ldots & p_{2N1} \\ p_{311} & \ldots & p_{3N1} \\ p_{411} & \ldots & p_{4N1} \end{pmatrix} \end{array} \quad S_2 = \begin{array}{c} \\ t_1 \\ t_2 \\ t_3 \\ t_4 \end{array} \begin{array}{c} \begin{array}{ccc} a_1 & \ldots & a_N \end{array} \\ \begin{pmatrix} p_{112} & \ldots & p_{1N2} \\ p_{212} & \ldots & p_{2N2} \\ p_{312} & \ldots & p_{3N2} \\ p_{412} & \ldots & p_{4N2} \end{pmatrix} \end{array} \quad S_3 = \begin{array}{c} \\ t_1 \\ t_2 \\ t_3 \\ t_4 \end{array} \begin{array}{c} \begin{array}{ccc} a_1 & \ldots & a_N \end{array} \\ \begin{pmatrix} p_{113} & \ldots & p_{1N3} \\ p_{213} & \ldots & p_{2N3} \\ p_{313} & \ldots & p_{3N3} \\ p_{413} & \ldots & p_{4N3} \end{pmatrix} \end{array} \tag{4.4}$$

4.2 Bootstrapping (Historical Simulation)

In addition to the Monte Carlo simulation, a bootstrapping method is used to generate scenarios. Like the Monte Carlo simulation, the bootstrapping takes care of the correlation between the several assets. The historical daily returns[14] of the assets $a_1, \ldots, a_N$ in the matrix R.

$$R = \begin{array}{c} \\ t_1 \\ t_2 \\ t_3 \\ t_4 \end{array} \begin{array}{c} \begin{array}{cccc} a_1 & a_2 & \ldots & a_N \end{array} \\ \begin{pmatrix} r_{11} & r_{12} & \ldots & r_{1N} \\ r_{21} & r_{22} & \ldots & r_{2N} \\ r_{31} & r_{32} & \ldots & r_{3N} \\ r_{41} & r_{42} & \ldots & r_{4N} \end{pmatrix} \end{array} \tag{4.5}$$

will be randomly re-sampled to the new matrix R' in which the asset returns of a single day from R will be copied as whole row into R' so that the correlation between the different asset returns remains the same.

$$R' = \begin{array}{c} \\ t_3 \\ t_2 \\ t_4 \\ t_1 \end{array} \begin{array}{c} \begin{array}{cccc} a_1 & a_2 & \ldots & a_N \end{array} \\ \begin{pmatrix} r_{31} & r_{32} & \ldots & r_{3N} \\ r_{21} & r_{22} & \ldots & r_{2N} \\ r_{41} & r_{42} & \ldots & r_{4N} \\ r_{11} & r_{12} & \ldots & r_{1N} \end{pmatrix} \end{array} \tag{4.6}$$

To calculate the prices in a convenient way, we add a new row of 1's at the top of R' and add 1 to all returns in the remaining rows. This gives the matrix R''.

$$R'' = \begin{array}{c} \\ t_0 \\ t_3 \\ t_2 \\ t_4 \\ t_1 \end{array} \begin{array}{c} \begin{array}{cccc} a_1 & a_2 & \ldots & a_N \end{array} \\ \begin{pmatrix} 1 & 1 & \ldots & 1 \\ 1+r_{31} & 1+r_{32} & \ldots & 1+r_{3N} \\ 1+r_{21} & 1+r_{22} & \ldots & 1+r_{2N} \\ 1+r_{41} & 1+r_{42} & \ldots & 1+r_{4N} \\ 1+r_{11} & 1+r_{12} & \ldots & 1+r_{1N} \end{pmatrix} \end{array} \tag{4.7}$$

The path of the asset prices are then calculated by the cumulative products of the columns of R''. The result is a price matrix S, which has the form:

[14] where $t_1, \ldots, t_4$ represent 4 trading days.

$$S = \begin{array}{c} \\ t_0 \\ t_1 \\ t_2 \\ t_3 \\ t_4 \end{array} \begin{array}{c} \begin{array}{cccc} a_1 & a_2 & \dots & a_N \end{array} \\ \begin{pmatrix} p_{01} & p_{02} & \dots & p_{0N} \\ p_{11} & p_{12} & \dots & p_{1N} \\ p_{21} & p_{22} & \dots & p_{2N} \\ p_{31} & p_{32} & \dots & p_{3N} \\ p_{41} & p_{42} & \dots & p_{4N} \end{pmatrix} \end{array} \tag{4.8}$$

From the price matrix S, the returns over the time period T, are calculated and stored to the matrix R''' in the form:

$$R''' = \begin{array}{c} \\ s_1 \\ s_2 \\ s_3 \\ \vdots \\ s_J \end{array} \begin{array}{c} \begin{array}{cccc} a_1 & a_2 & \dots & a_N \end{array} \\ \begin{pmatrix} r_{11} & r_{12} & \dots & r_{1N} \\ r_{21} & r_{22} & \dots & r_{2N} \\ r_{31} & r_{32} & \dots & r_{3N} \\ \vdots & \vdots & \ddots & \vdots \\ r_{J1} & r_{J2} & \dots & r_{JN} \end{pmatrix} \end{array} \tag{4.9}$$

where $r_{11}, \dots, r_{JN}$ represent the returns of the time period T, in this case the return for 4 trading days, for the different simulation runs $s_1, \dots, s_J$. The corresponding implementation in Matlab of the bootstrapping approach looks as follows:

```
function [periodReturns] = bootstrap(R, nDays, nSims)
    nAssets = size(R,2);
    periodReturns = nan(nSims, nAssets);
    for iSim = 1:nSims
        dReturns = [ones(1, nAssets); nan(nDays, nAssets)];
        for day = 1:nDays
            pos = randi(length(R));
            dReturns(day+1, :) = R(pos, :)+1;
        end
        prices = cumprod(dReturns);
        periodReturns(iSim, :) = (prices(end,:) - prices(1,:))./prices(1,:);
    end
end
```

where `R` contains the historical daily returns for each asset, `nDays` is the number of days to bootstrap returns, and `nSims` is the number of simulations. The method returns the matrix R''', the returns over the time period T (`periodReturns`). This matrix can be directly used as input of the optimization approach, which was elaborated in Chapter 3.

Chapter 5

Analysis of the Implemented Optimization

The elaborated and implemented approach of Rockafellar and Uryasev [20] in Chapter 3 is now used to optimize a portfolio composed of a set of assets from different asset classes. This chapter shows the composition of the portfolio, the calibration of the simulation methods, and the observations of the optimization. To analyze the implemented approach, the results are compared with the output of a brute-force optimization. The analysis procedure is illustrated in Figure 5.1.

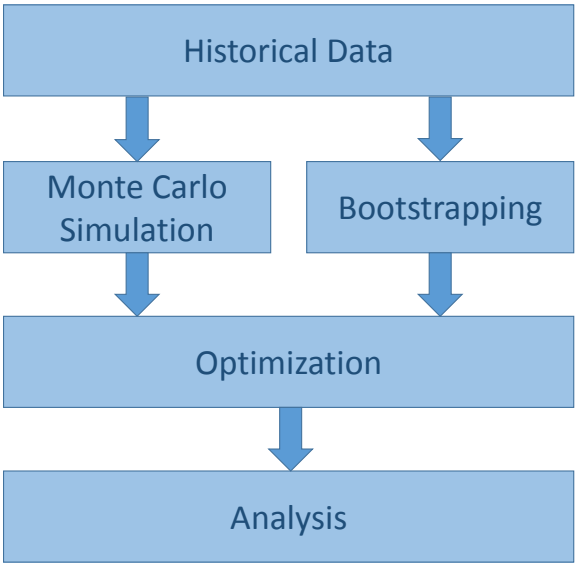

Figure 5.1: Firstly, the historical data of the underlying assets in the portfolio will be analyzed and used for the simulations. Secondly, the generated scenarios will be optimized and analyzed.

5.1 Portfolio Composition

To provide a wide range of diversification the portfolio contains US and global indices from different asset classes, as shown in Table 5.1. The historical market data is sourced from Datastream[15] in the form of daily prices.

Table 5.1: The portfolio is composed of several assets from different asset classes. The corresponding daily prices are retrieved from Thomson Reuters Datastream.

Asset	Asset Class	Description	Datastream Id
CASH	Cash	Barclays US Short Treasury	LHSHORT(IN)+100
CORB	Fixed Income	Barclays US Credit 5-10Y (Corp. Bonds)	LHMFI10(IN)+100
GOVB	Fixed Income	Barclays US Gov. Bonds	LHGOVBD(IN)+100
GOLD	Commodities	Handy & Harman Gold USD (Troy Oz)	GOLDHAR
COMM	Commodities	CRB BLS Spot Index - Price Index	CRBSPOT
MSCI	Equity	MSCI Global Equity Index	MSWRLDL(MSPI)
SP500	Equity	S&P 500 US Equity Index	S&PCOMP(PI)

5.2 Scenario Generation

To test the implemented optimization approach with two different scenario generating methods, future asset prices are simulated both by Monte Carlo simulation from a model and by bootstrapping from historical returns.

5.2.1 Market Data Analysis

Since the generated future asset prices should rely on the history of the assets in the portfolio, the underlying historical market data (see Section 5.1) need to be analyzed and calibrated. This means that the time period has to be defined. As a rule of thumb for reliable and reasonable statistical estimations, the number of trading days to be analyzed has to be between 5 to 20 times the number of the assets in the portfolio. Because the portfolio is composed of seven assets, the number of trading days is defined as 126, which is the half of the number of trading days per year (252 / 2 = 126). The Datastream market data (provided by Credit Suisse) is available until the end of the year 2013. Hence, we analyze the historical prices of the assets in the portfolio between July 1 and December 31, 2013, visualized in Figure 5.2, to get the required input for the simulations. As demonstrated in Section 4.1, the Monte Carlo simulation is based on multivariate geometric Brownian motion, the expected return μ, the standard deviation σ, and

[15] Thomson Reuters Datastream is a provider of financial market data.

the correlation ρ are required to generate correlated asset paths (see Table 5.2 and Table 5.3). In addition, VaR and CVaR are also measured.

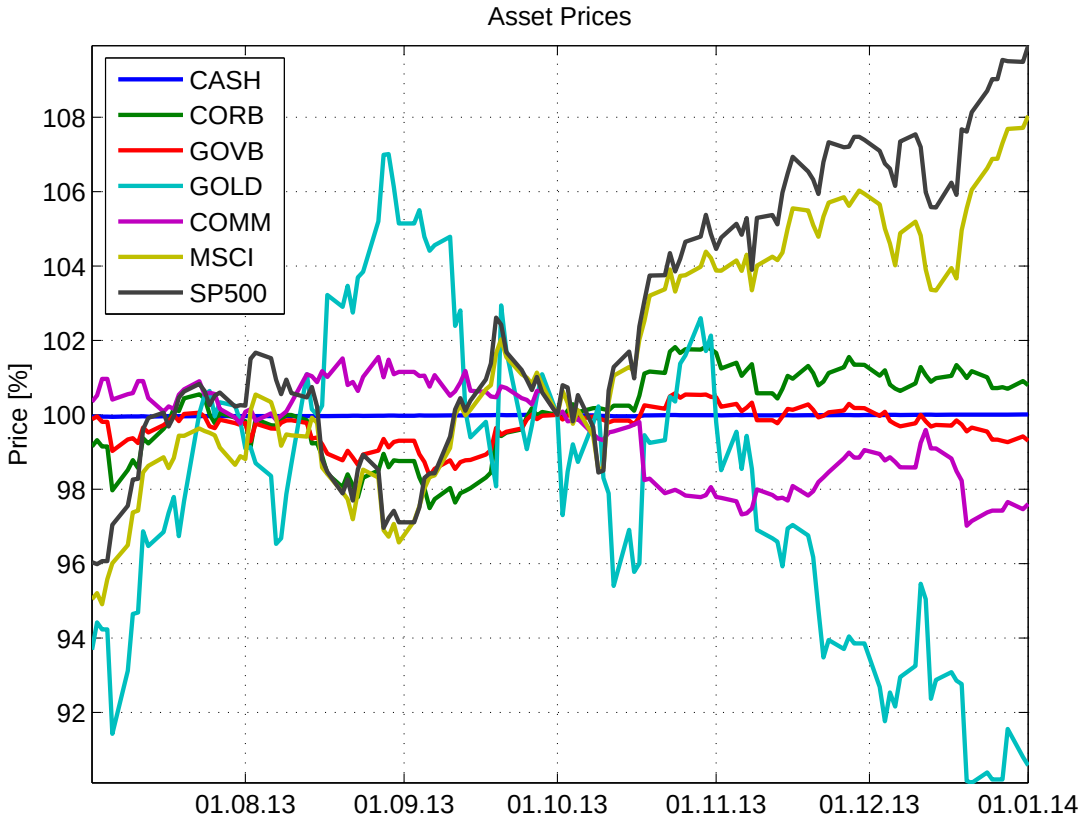

Figure 5.2: Prices of the underlying assets in the portfolio measured between July 1 and December 31, 2013. The prices are normalized to 100% at midpoint of the time period.

Table 5.2: Measured historical daily returns for the underlying assets in the portfolio in the time period between July 1 and December 31, 2013 for the annualized expected return μ, the annualized standard deviation σ, VaR, and CVaR at the confidence level $\beta = 95\%$ in the specified time period.

Asset	$\boldsymbol{\mu}$[%]	$\boldsymbol{\sigma}$[%]	**VaR** [%]	**CVaR** [%]
CASH	0.11	0.06	-0.00	-0.01
CORB	4.69	4.63	-0.43	-0.59
GOVB	-0.03	2.89	-0.26	-0.40
GOLD	-3.67	19.45	-2.27	-2.63
COMM	-5.84	5.04	-0.53	-0.87
MSCI	22.93	8.18	-0.76	-1.03
SP500	24.34	9.65	-0.86	-1.27

Table 5.3: Measured historical daily returns for the correlation between the underlying assets in the portfolio in the time period between July 1 and December 31, 2013. The correlation is needed to generate correlated asset paths with geometric Brownian motion (see Section 4.1.2).

	CASH	**CORB**	**GOVB**	**GOLD**	**COMM**	**MSCI**	**SP500**
CASH	1.0000	0.1961	0.1759	0.2558	-0.1619	0.0736	0.0689
CORB	0.1961	1.0000	0.9755	0.2631	-0.1713	0.0605	0.0794
GOVB	0.1759	0.9755	1.0000	0.2164	-0.1708	-0.0383	0.0119
GOLD	0.2558	0.2631	0.2164	1.0000	0.0824	0.0568	0.0206
COMM	-0.1619	-0.1713	-0.1708	0.0824	1.0000	0.0202	0.0084
MSCI	0.0736	0.0605	-0.0383	0.0568	0.0202	1.0000	0.9161
SP500	0.0689	0.0794	0.0119	0.0206	0.0084	0.9161	1.0000

5.2.2 Simulated Trading Days and Number of Simulations

The number of simulations are defined as 1'000 and the scenarios are generated for 10 trading days. The arithmetic returns are calculated as the difference of the last price to the initial price divided by the initial price, as described in Equation 2.1, with the function `simpaths2ret` (see Appendix A).

5.2.3 Monte Carlo Simulation

The results of the 1'000 simulated future 10-day returns from the Monte Carlo simulation can be observed in Table 5.4. The measurements show that there are marginal differences to the historical return distribution. This is related to the relatively small number of simulation runs. By increasing the number of simulation runs, the difference between the simulated return distribution and the historical return distribution, including the difference between the correlations, will be diminished.

Table 5.4: Measures of the 1'000 simulated 10-day returns for the underlying assets in the portfolio for the annualized expected return μ, the annualized standard deviation σ, VaR and CVaR at the confidence level $\beta = 95\%$ for a period of 10 trading days.

Asset	$\boldsymbol{\mu}$[%]	$\boldsymbol{\sigma}$[%]	**VaR** [%]	**CVaR** [%]
CASH	0.11	0.06	-0.02	-0.02
CORB	4.89	4.54	-1.30	-1.65
GOVB	0.15	2.84	-0.93	-1.14
GOLD	-7.90	18.88	-6.25	-7.66
COMM	-5.63	5.03	-1.84	-2.20
MSCI	22.54	7.89	-1.69	-2.38
SP500	22.54	9.48	-2.23	-2.91

5.2.4 Bootstrapping

The second method to simulate future 10-day returns, is done by the bootstrapping from historical returns, as described in Section 4.2. Like for the Monte Carlo simulation, the historical returns of Section 5.2.1 serve as basis. The difference is that the expected return μ, the standard deviation σ and the correlation matrix ρ do not have to be measured from the historical returns. As shown in Table 5.5, the results of 1'000 bootstrapping runs are similar to the Monte Carlo simulation.

Table 5.5: Measures of the 1'000 simulated 10-day returns for the underlying assets in the portfolio for the annualized expected return μ, the annualized standard deviation σ, VaR and CVaR at the confidence level $\beta = 95\%$ for a period of 10 trading days.

Asset	$\boldsymbol{\mu}$**[%]**	$\boldsymbol{\sigma}$**[%]**	**VaR [%]**	**CVaR [%]**
CASH	0.12	0.06	-0.02	-0.02
CORB	4.25	4.76	-1.39	-1.79
GOVB	-0.23	2.97	-0.98	-1.24
GOLD	-5.88	18.64	-5.99	-7.16
COMM	-4.66	4.97	-1.91	-2.43
MSCI	21.88	8.07	-1.86	-2.51
SP500	24.45	9.44	-2.05	-2.70

5.3 Optimization with CVaR Constraints

The simulated 10-day returns, generated in Section 5.2, are further used to optimize the portfolio, which is composed of the assets described in Section 5.1. Running the optimization algorithm with several CVaR limits, according to the 10-day return distribution, results in different asset allocations. The measures in Table 5.6 and Table 5.7 with its corresponding illustrations in Figure 5.3 and Figure 5.4 show that by decreasing the risk level, the weight of the risky assets in the portfolio is reduced by the optimization algorithm. This preference of less risky assets imply a lower expected return of the portfolio.

By comparing the results of the optimization of the Monte Carlo and the bootstrapping generated scenarios, one can see that the risk and return characteristic of the optimized portfolio has only marginal differences. This fact emphasizes the stability of the optimization approach of Rockafellar and Uryasev [20] and its advantage to handle any kind of return distribution. However, due to the different generated return scenarios, the optimization algorithm allocates the assets in the portfolio differently.

Table 5.6: Based on Monte Carlo scenarios, five optimization runs with different CVaR limits, according to the 10-day return distribution for a confidence level of 95%, result in different asset allocations, VaR's (for 10-day return distribution and a confidence level of 95%), annualized standard deviations, and annualized expected returns.

CVaR Limit [%]	-2.50	-2.00	-1.50	-1.00	-0.50
Exp. Return [%]	22.52	18.64	13.92	9.01	4.15
Std. Dev. [%]	7.89	6.32	4.69	3.02	1.38
VaR [%]	-1.75	-1.34	-1.00	-0.66	-0.32
CASH [%]	0.01	0.07	17.36	47.34	76.78
CORB [%]	0.06	21.93	26.70	16.43	6.57
GOVB [%]	0.01	0.03	0.03	0.03	0.03
GOLD [%]	0.01	0.01	0.01	0.01	0.01
COMM [%]	0.01	0.01	0.01	0.01	0.01
MSCI [%]	99.81	77.91	55.85	36.14	16.56
SP500 [%]	0.09	0.04	0.04	0.04	0.04

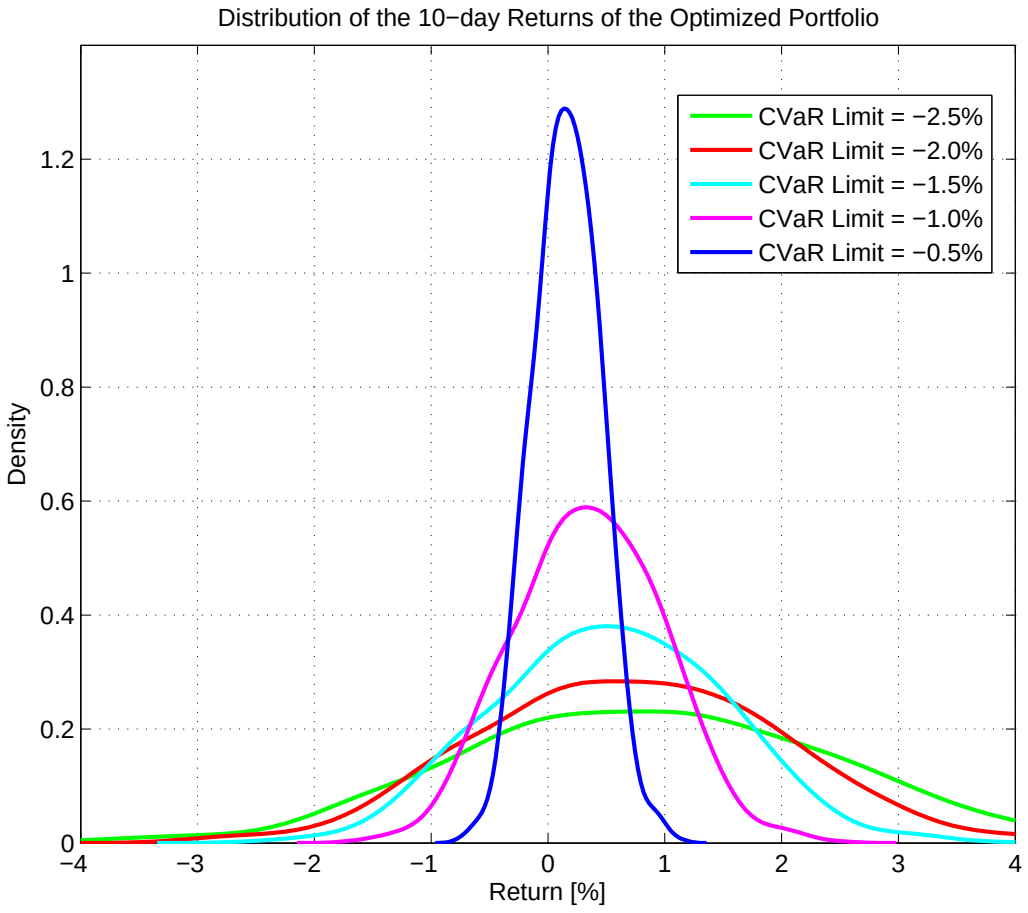

Figure 5.3: Based on Monte Carlo scenarios, five optimization runs with different CVaR limits, according to the 10-day return distribution for a confidence level of 95%, result in different 10-day return distributions. The lower the risk limit is, the more narrow is the return distribution. Additionally, the expected returns decrease with more restrictive risk limits.

Table 5.7: Based on bootstrapped scenarios, five optimization runs with different CVaR limits, according to the 10-day return distribution for a confidence level of 95%, result in different asset allocations, VaR's (for 10-day return distribution and a confidence level of 95%), annualized standard deviations, and annualized expected returns.

CVaR Limit [%]	-2.50	-2.00	-1.50	-1.00	-0.50
Exp. Return [%]	22.30	18.32	13.93	8.91	4.13
Std. Dev. [%]	8.35	6.60	4.95	3.15	1.45
VaR [%]	-1.80	-1.41	-1.08	-0.71	-0.34
CASH [%]	0.02	0.04	15.64	45.01	74.22
CORB [%]	8.60	27.21	30.07	20.57	10.12
GOVB [%]	0.02	0.03	0.03	0.03	0.03
GOLD [%]	0.01	0.02	0.02	0.03	0.02
COMM [%]	0.01	0.01	0.01	0.02	0.02
MSCI [%]	15.63	23.88	24.27	16.15	7.48
SP500 [%]	75.71	48.81	29.96	18.19	8.11

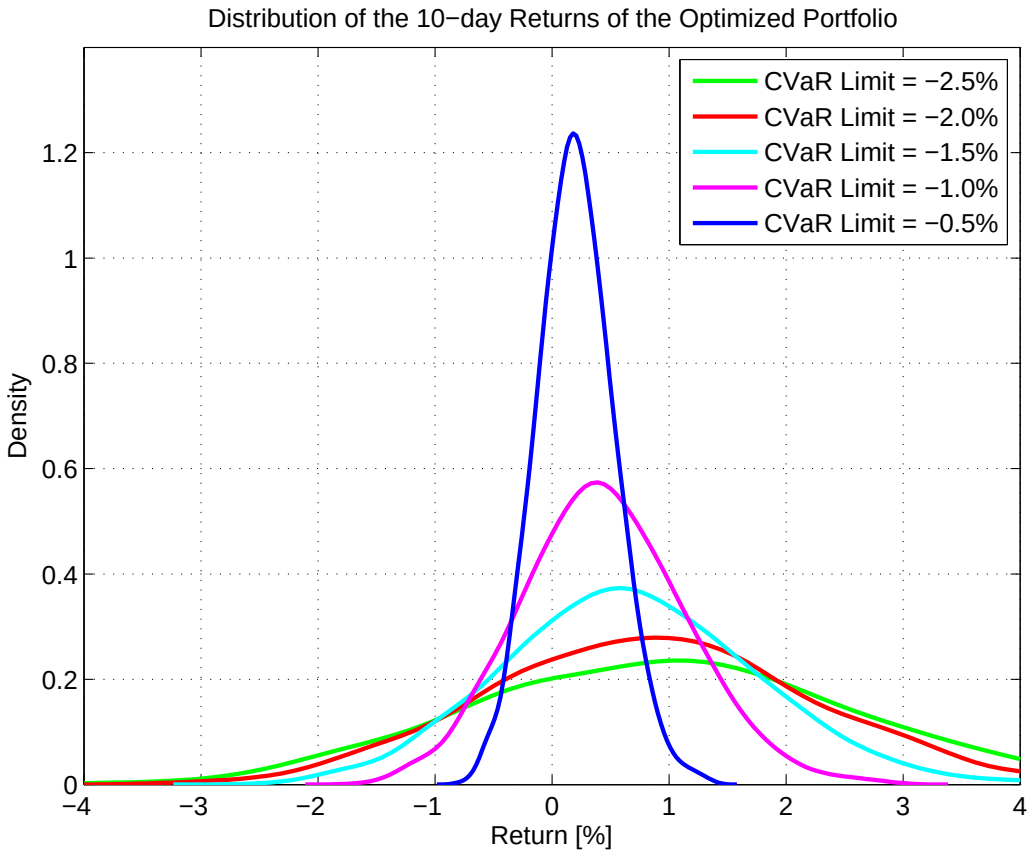

Figure 5.4: Based on bootstrapped scenarios, five optimization runs with different CVaR limits, according to the 10-day return distribution for a confidence level of 95%, result in different 10-day return distribution. The lower the risk limit is, the more narrow is the return distribution. Additionally, the expected returns decrease with more restrictive risk limits.

In Figure 5.5, Figure 5.6 and Figure 5.7 the optimization path of the asset weights for different CVaR levels is visualized. The search starts with an equally weighted portfolio as it was specified with `w0` in Section 3.3.2.5. The optimization iteration is the number of steps until the algorithm finds the maximized expected return with the specified CVaR risk constraints. As shown in Table 5.6 and Table 5.7, the search path visualization also demonstrates that with increasing the risk level, the riskier assets, in this case the equity assets, are allocated more. If there is a lower bound defined, as described in Section 3.3.2.3, the optimization tries to find a solution with respect to these additional constraints. In Figure 5.7 the consequence of a lower bound constraint of 5% for all assets in the portfolio is visualized. The CVaR levels of -0.5% and -1.5% for the 10-day return distribution of the bootstrapped scenarios are chosen to demonstrate the optimization path.

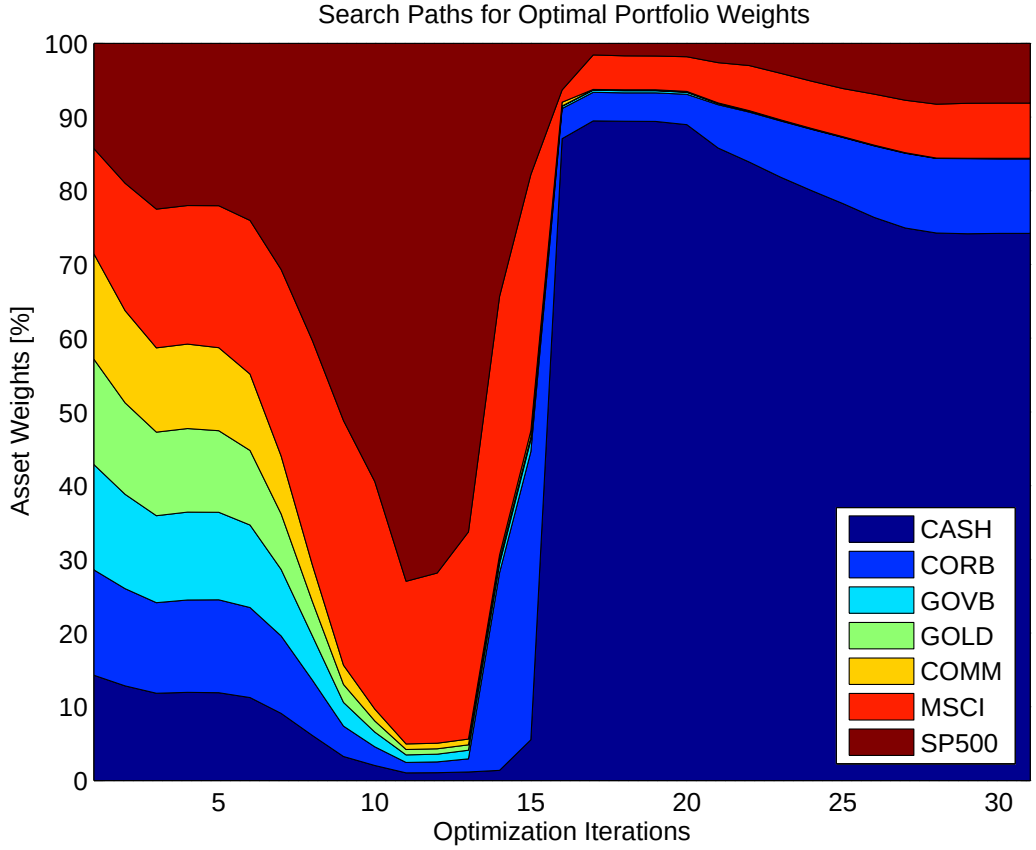

Figure 5.5: The search path visualizes how the algorithm finds the optimal weights for the maximum expected return with the given constraints. In this case, the CVaR level of -0.5% of the 10-day return distribution, based on the scenarios of the bootstrapping method, is shown. Because of the low risk level, mainly cash is allocated.

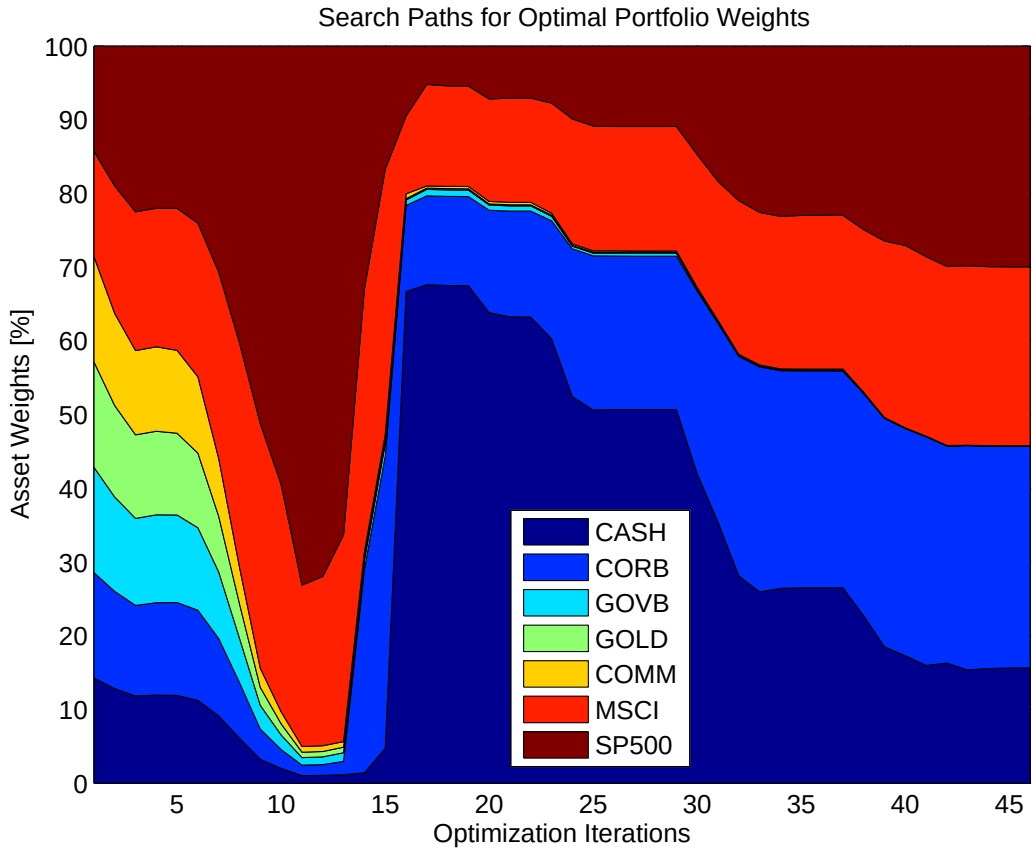

Figure 5.6: The search path visualizes how the algorithm finds the optimal weights for the maximum expected return with the given constraints. In this case, the CVaR level of -1.5% of the 10-day return distribution, based on the scenarios of the bootstrapping method, is shown. Because the risk level is higher than in Figure 5.5, only 15% of the portfolio is allocated to cash. The equity assets (MSCI, SP500) and the corporate bond asset (CORB) share the majority of the remaining portfolio weight.

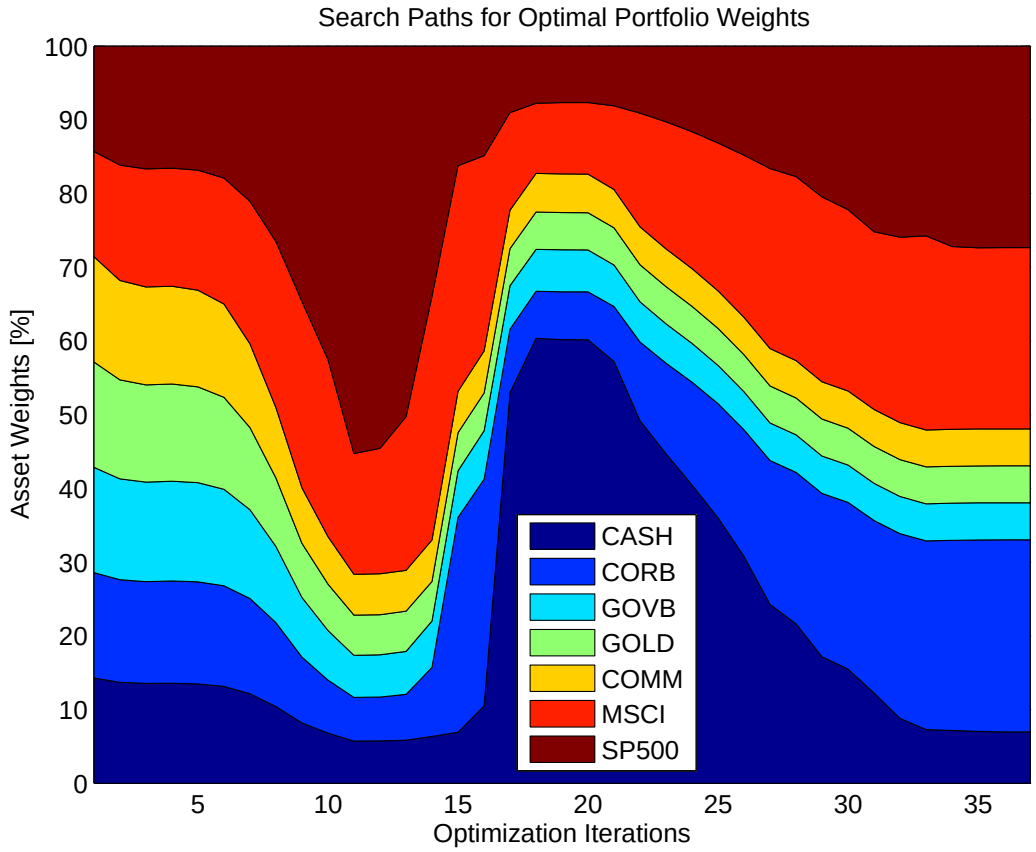

Figure 5.7: The search path visualizes how the algorithm finds the optimal weights for the maximum expected return with the given constraints. In this case, the CVaR level of -1.5% of the 10-day return distribution, based on the scenarios of the bootstrapping method, is shown. In this scenario, there is a lower bound constraint of %5 for all assets. This means that the algorithm needs to allocate a minimum of %5 for an asset in the portfolio.

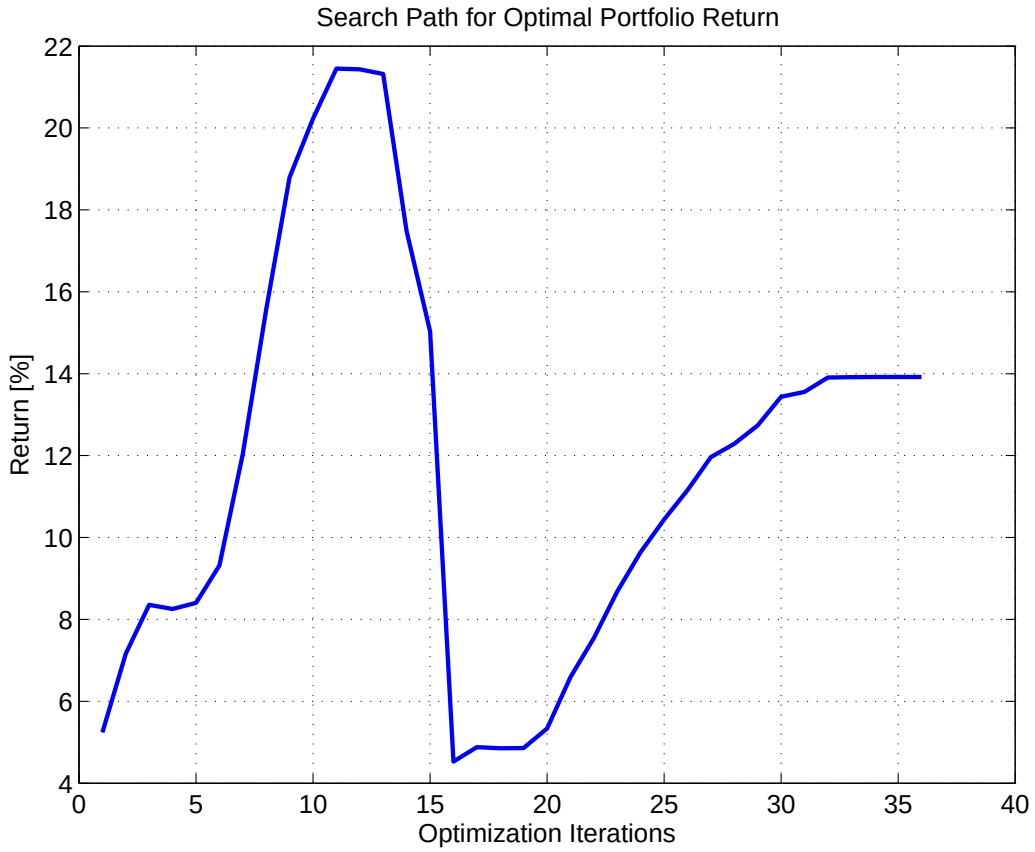

Figure 5.8: The optimization path of the annualized expected return for the CVaR level of -1.5% of the 10-day return distribution and a confidence level of 95%.

In Figure 5.8 the optimization path of the expected return for the CVaR level of -1.5% of the 10-day return distribution and a confidence level of 95% is visualized. One can see that the algorithm starts with a relatively low value, increases the value, decreases it, and increases it again until the maximum expected return is found after about 35 iterations. By comparing the search paths for the asset weights, one can see that the search path of the expected value has the opposite deviation. The reason for this is that the algorithm allocates more equity, and thus more cash during the several iteration in the optimization search path, which directly influences the value of the expected return.

As demonstrated in Figure 5.9, when the simulation and the optimization for a specified risk level is run multiple times, the optimization algorithm allocates the assets so that the defined CVaR limit is not exceed. The equally weighted portfolios are not only more risky, but they also have lower expected returns.

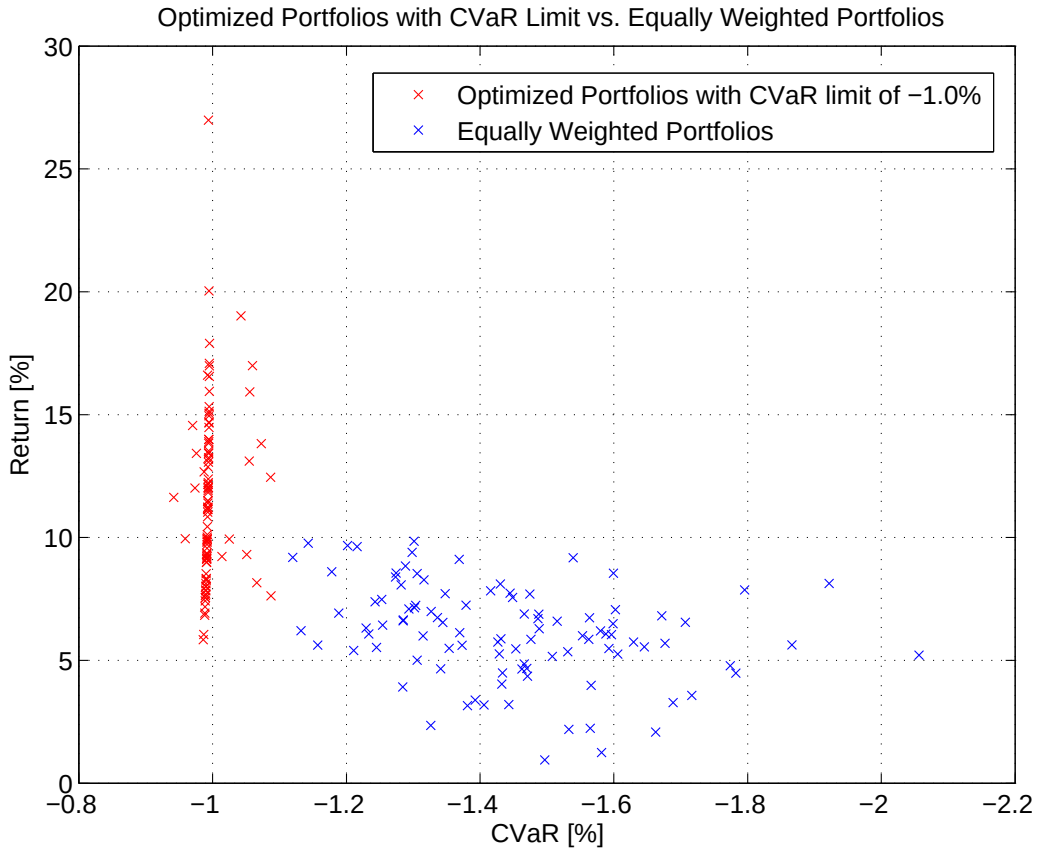

Figure 5.9: Comparison between optimized portfolios (red) and equally weighted portfolios (blue). The optimization algorithm allocates portfolios so that a maximum expected return is the objective with the constraint that a certain risk limit is not exceeded. In this case the risk limit, defined as CVaR of -1.0% of the 10-day return distribution with a confidence level of 95%. The return is the annualized mean of the expected 10-day return.

5.4 Brute-Force Optimization

To compare and analyze the results from the CVaR optimization approach, a brute-force approach is used in which a grid of weight combinations are generated to allocate the scenarios. The generation of the weight combination is implemented in Matlab with the function `weights`. It returns a matrix `W` with the dimensions `nCombinations`-by-`nAssets` (see Equation 5.1). The rows are defined by $w_{ci}, \ldots, w_{CN}$ where c is the combination index, C the number of combinations, and N the number of assets. Every row has a sum of 1.

The Matlab implementation of the `weights` functions is:

```
function W = weights(nAssets, nCombinations)
    W=rand(nCombinations, nAssets);
    rowsums=repmat(sum(W, 2), 1, size(W, 2));
    W=W./rowsums;
end
```

which returns a grid of weight combinations W in the form:

$$W = \begin{array}{c} \\ c_1 \\ c_2 \\ \vdots \\ c_C \end{array} \begin{array}{c} \begin{array}{cccc} a_1 & a_2 & \dots & a_N \end{array} \\ \begin{pmatrix} w_{11} & w_{12} & \dots & w_{1N} \\ w_{21} & w_{22} & \dots & w_{2N} \\ \vdots & \vdots & \ddots & \vdots \\ w_{C1} & w_{C2} & \dots & w_{CN} \end{pmatrix} \end{array} \tag{5.1}$$

Like the optimization approach of Rockafellar and Uryasev, the brute-force method requires generated scenarios. We use the Monte Carlo simulation, described in Section 5.2.3, which generates 1'000 10-day arithmetic returns for each asset in the portfolio. The portfolios P are then constructed by the matrix multiplication of the weight combinations W and the returns r:

$$P = r * W^T \tag{5.2}$$

In the matrix P in which every column represents a portfolio that weights the returns r to the according weight combination of W. P has the form:

$$P = \begin{array}{c} \begin{array}{cccc} P_1 & P_2 & \dots & P_C \end{array} \\ \begin{pmatrix} r_{11} & r_{21} & \dots & r_{C1} \\ r_{12} & r_{22} & \dots & r_{C2} \\ \vdots & \vdots & \ddots & \vdots \\ r_{1N} & r_{2N} & \dots & r_{CN} \end{pmatrix} \end{array} \tag{5.3}$$

The portfolios, generated with the brute-force method, are represented in Figure 5.10 with annualized expected return and CVaR for the 10-day return distribution and a confidence level of 95%. The colored circles mark the portfolios with the highest annualized expected return for the different CVaR levels, defined in Table 5.7 and Table 5.7. The portfolios are on the efficient frontier and their expected return is similar to the portfolio allocations, optimized with the approach of Rockafellar and Uryasev. However, because the expected return of the brute-force

generated portfolios is lower, one can see that the Rockafellar and Uryasev optimization works more efficient and with better results.

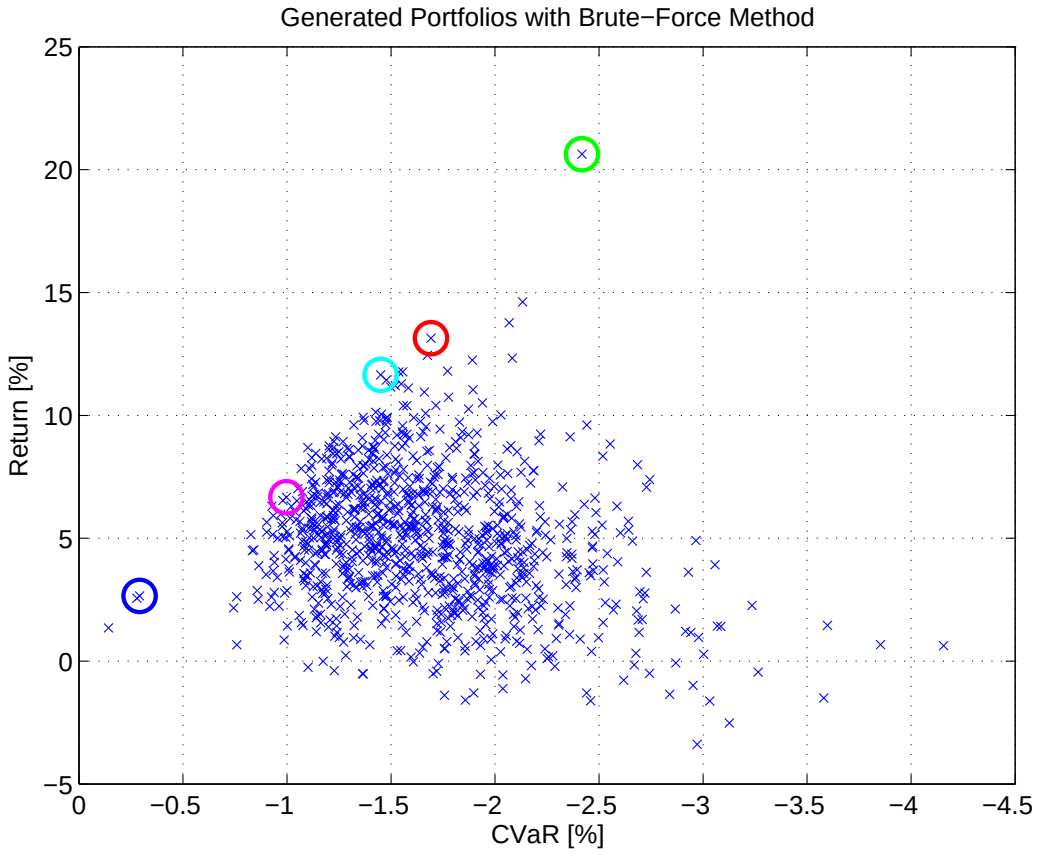

Figure 5.10: Brute-force generated portfolios with annualized expected return and CVaR for the 10-day return distribution for the confidence level of 95%. The colored circles mark the portfolios with the highest annualized expected return for the different CVaR levels, defined in Table 5.7 and Table 5.7.

By analyzing the 10-day return distribution for different CVaR limits, one can see in Figure 5.11 that the brute-force optimization has similar results to the Rockafellar and Uryasev optimization. Low risk limits have a more narrow return distribution than portfolios with a higher risk limit, and the expected return grows when then risk limit is increased.

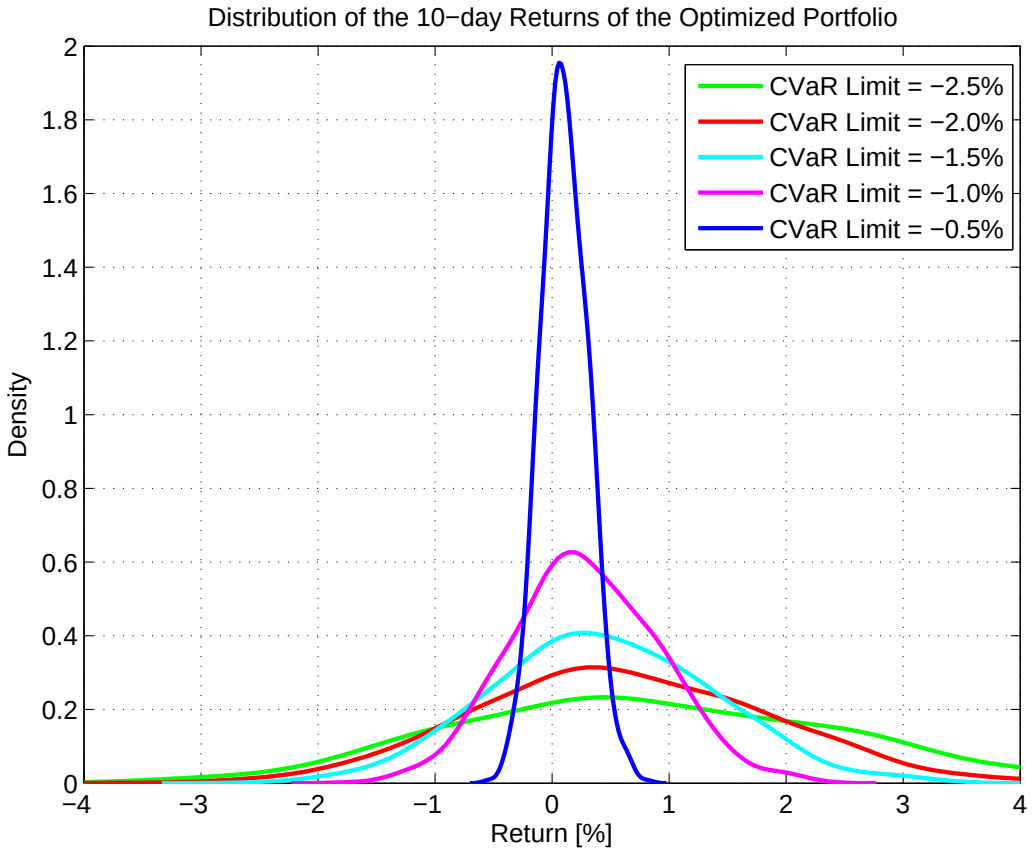

Figure 5.11: Based on Monte Carlo scenarios, a brute-force optimization with different CVaR limits, according to the 10-day return distribution for a confidence level of 95%, results in different 10-day return distributions. The lower the risk limit is, the more narrow is the return distribution. Additionally, the expected returns decrease with more restrictive risk limits.

Chapter 6

IT Architecture

From a global bank perspective, not only the implementation of the portfolio optimization but also the integration in an IT system landscape, which supports a sophisticated investment process, is of major concern. This chapter focuses on how the investment process conceptually works and how the portfolio optimization, elaborated and implemented in Chapter 3, can be integrated into this investment process from an IT architecture perspective.

6.1 Investment Process

Financial portfolios of private or institutional clients[16] are constructed and maintained within a structured investment process, as illustrated in Figure 6.1, containing the following steps:

1. Gathering Client Requirements
2. Strategic Asset Allocation: Definition of benchmarks with long-term assumptions (1-5 years)
3. Tactical Asset Allocation: Allocation of markets and generic asset classes in short-term views (3–6 months)
4. Portfolio Management: Selection and optimization of investment instruments (monthly, weekly or daily)
5. Reporting and Monitoring

[16]Institutional clients are organizations which pool large sums of money and invest those sums in investment assets. Typically institutional investors are banks, insurance companies, pension funds, hedge funds and mutual funds.

Figure 6.1: A sophisticated investment process is divided into several steps, which cover the gathering of client requirements, a strategic asset allocation, a tactical asset allocation, a portfolio management (portfolio engineering), and reporting and monitoring. Source: Credit Suisse [25].

It is a philosophic question which step of the investment process the most responsible for the portfolio performance is. Studies show that the performance is mainly driven by strategic and tactical portfolio allocation (step 2 and 3) [23]. However, a portfolio is composed of instruments and, therefore, the portfolio engineering (step 4) needs an elaborated portfolio optimization method to control the risks. How the bank's investment process should look like and which tools and methods are used, needs to be defined on a strategic bank level, and is the responsibility of the chief investment officer (CIO)[17].

6.2 High Level Architecture

Next, we consider an IT architecture for the portfolio management (step 4 in Section 6.1). This includes a description of the portfolio management process, a high level system overview, and a description of the interfaces for the corresponding down- or upstream systems[18].

6.2.1 Portfolio Management Process

The IT system, as shown in Figure 6.2, reflects the process of portfolio management where investment analysts enter their forecasts (e.g., capital-market assumptions, key drivers, and mega trends) into the system. The system provides several forms for the different forecasts factors[19] which is typically updated after investment committee meetings (e.g., every two weeks). Relationship managers have the task to inform, discuss, and agree together with the client investment objectives, risk tolerance, and the desired investment universe.

[17] The chief investment officer usually oversees a team of professionals that has responsibilities such as managing and monitoring investment activity, working with analysts and researchers. They develop short-term and long-term investment policies.

[18] From an application's perspective, upstream data flows away from the application. Conversely, downstream traffic flows into the application.

[19] Forecasts factors can be used to manage market forecasts, e.g., for equity, fixed-income

Finally, portfolio managers construct and maintain portfolios according to the inputs of the investment analysts and the relationship managers. They need to take care that strategic and tactical investment specifications, defined by the chief investment office, are considered. After all required information is entered, the portfolios are ready to be optimized either by a mean-variance approach or by the shortfall risk optimization approach of Rockafellar and Uryasev, implemented in Chapter 3. Both optimization approaches use the entered information of investment analysts, relationship managers, portfolio managers as well as the historical market data of underlying assets in the portfolio to calculate an optimized asset allocation.

The selection of the optimization approach depends on the preference of the portfolio managers, who can also optimize portfolios with both optimization approaches and compare the asset allocations. The scenarios for the optimization can either be generated by a Monte Carlo simulation from a model or by bootstrapping from historical returns. As we have seen in Chapter 2 and Chapter 3, the optimization approach of Rockafellar and Uryasev can also handle non-normal return distributions where it prefers positive skewness, small kurtosis and low variance.

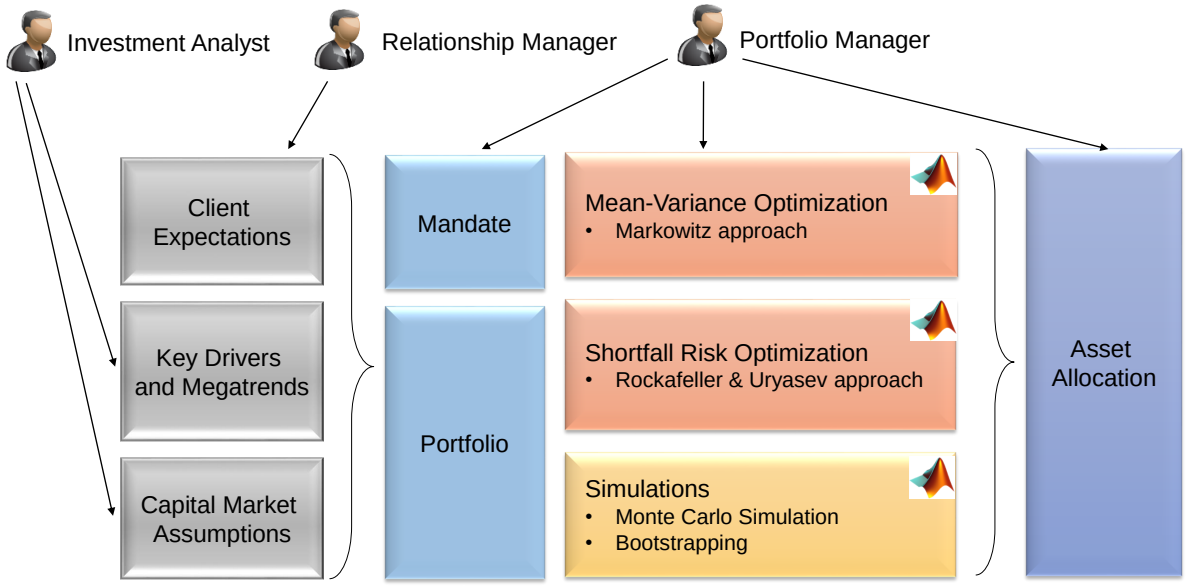

Figure 6.2: The portfolio management process is reflected in the target IT architecture. It provides a multi-user environment for investment analysts, relationship managers, and portfolio managers. The inputs are the client expectations, key drivers and mega trends, and capital market assumptions. The portfolio manager can choose between two optimization approaches (shortfall risk optimization and mean-variance optimization) to calculate optimal asset allocations. The simulation of future prices of the financial assets underlying the portfolio is either done with Monte Carlo simulation or with bootstrapping. Both simulation methods use historical market data as input.

6.2.2 System Overview

Figure 6.3 represents the system with its infrastructural components[20]. The application is structured into three layers: graphical user interface (GUI), business logic (BL), and data management (DM). Furthermore functional components allow the separation of concerns and a deployment according to the underlying infrastructural components.

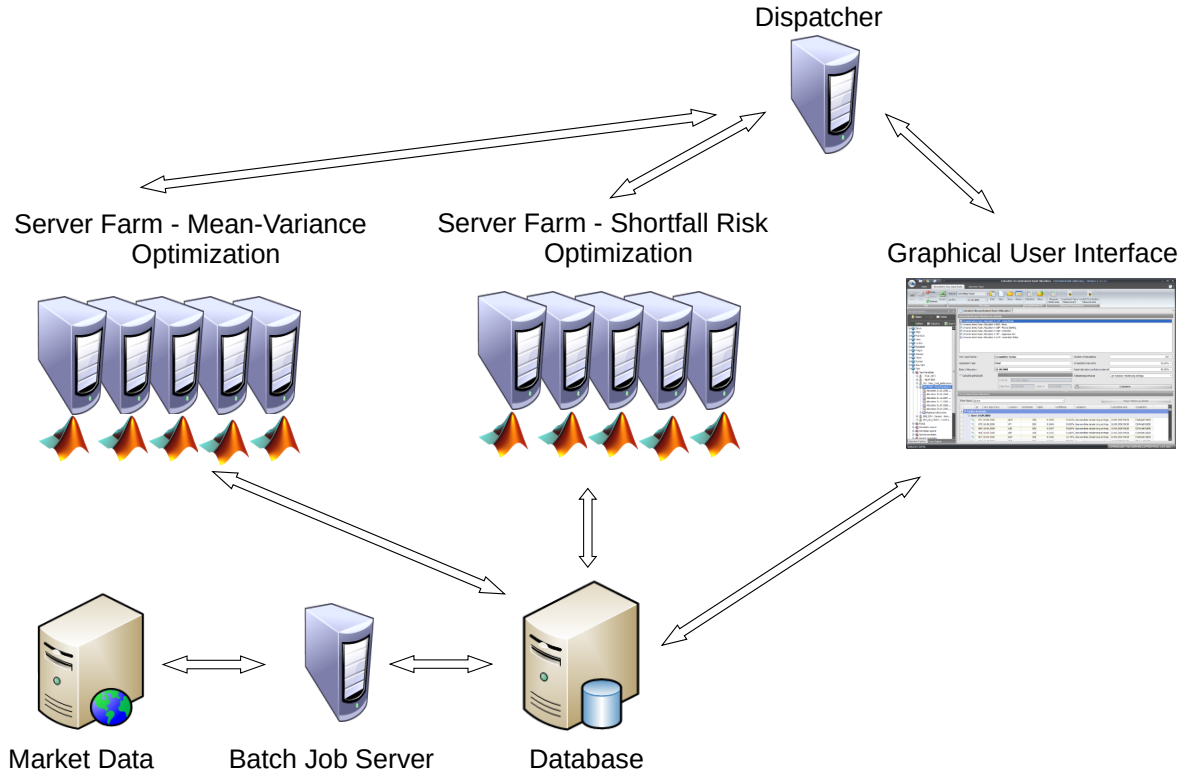

Figure 6.3: A dispatcher distributes optimization requests from the portfolio managers to the corresponding parallel computation server. A central database stores the market data, which is updated on a daily basis. It also stores the mandates, portfolios, and asset allocations which serve as input for the portfolio optimization, calculated on the corresponding server farm.

The design allows a server farm[21] per optimization approach (mean-variance and shortfall risk optimization). A dispatcher component distributes the incoming optimization requests from the portfolio managers to the specific parallel computing server. The implementation of the Rockafellar and Uryasev optimization with linear programming (see Section 3.2) allows the parallelization of the optimization problem. This means that the optimization algorithm can be deployed as a core component to several parallel computing servers with installed LP solvers (e.g., CPLEX, XPRESS). This allows the simultaneous optimization of large portfolios with a very large number of scenarios.

[20] Infrastructural components are application servers, databases, user computers, etc.
[21] Parallel computing application servers

The required data for the optimization is retrieved from a central database which contains investment forecasts, mandates, portfolios, constraints, historical market data. A batch job server ensures a daily update of market data of popular providers (e.g., Bloomberg, Thomson Datastream). In addition, the batch job server is also able to trigger nightly portfolio optimizations or the re-balancing of portfolios (e.g., at the end of month).

The GUI component is deployed on corresponding user computers. A role concept allows the investment analysts, relationship managers, and portfolio managers to use the same graphical user interface with different views and functionality. Relationship managers are responsible for the client and mandate settings, while investment analysts enter their forecasts about financial markets, and portfolio managers construct, maintain, and optimize their portfolios. Additionally, a user based security concept prevents the access to data of other users.

6.2.3 Interfaces

The bespoke system is connected to downstream and upstream systems. This means that several interfaces provide the exchange of information to and from the system (see Figure 6.4). In the context of the portfolio management system, the market data providers are downstream systems. Several web-service interfaces are used to retrieve market data by the different market data providers in order to store it in the central system database with a unified format. After the system has calculated the asset allocations, the information has to be sent by a system interface to the trading desk, which is responsible for the trading of the according instruments on a financial market (e.g., stock exchange). Furthermore, there is also an interface to export performance reports for clients or for quality and risk departments.

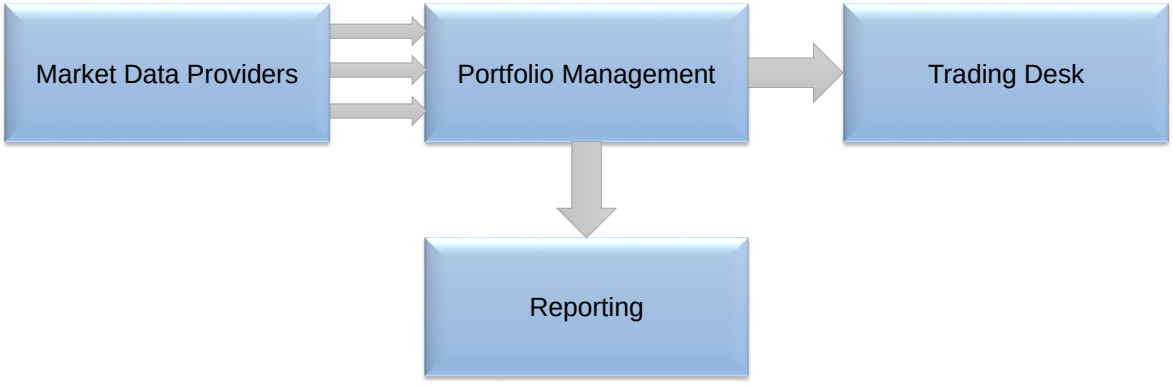

Figure 6.4: Interfaces allow the daily update of market data, retrieved from external market data providers. Further, interfaces can be used to export reports for clients or quality departments, and to send optimized asset allocations to the trading desk.

Chapter 7

Conclusion and Outlook

7.1 Conclusion

The elaboration of the approach of Rockafellar and Uryasev shows that the central element of their approach is a derived CVaR function which is independent of the VaR function. VaR has not to be calculated first, but it will be calculated and optimized simultaneously with the optimization of the asset allocation. Additionally, their CVaR function can be linearized which allows the implementation with linear programming techniques. This makes it a very efficient optimization method that is able to optimize very large portfolios with a large number of scenarios with relatively small computational resources. In contrast to the mean-variance approach, which penalizes returns equally to losses, the optimization approach of Rockafellar and Uryasev focuses on the shortfall risk which means it prefers a positive skewness, a small kurtosis, and a low variance.

It was demonstrated with two simulation methods that the approach of Rockafellar and Uryasev provides valid results. The examples have relatively low dimensions and are used for illustrative purposes. For the present thesis, the optimization was implemented with Matlab. However, because of the linear programming feature, it can also be implemented with a popular LP solver (e.g., Cplex, Xpress) which solves the the problem more efficiently. Since linear programming problems can be parallelized, a server farm, equipped with LP solvers, enables the optimization of a large number of portfolios with a large number of simulations simultaneously. The proposed IT architecture allows the integration of the elaborated approach of Rockafellar and Uryasev into an established investment process of a bank. The demonstrated target system can be used by different user roles to provide a defined process of portfolio management with mean-variance as well as with the optimization approach of Rockafellar and Uryasev. With its defined workflow and interfaces the system architecture allows a daily update of market data, sending

the optimized portfolio allocations to the trading desk, and generate specified reports for clients as well as for quality departments.

7.2 Outlook

The current thesis serves as an implementation guide for portfolio optimization for constrained shortfall risks with linear programming techniques. As shown, the optimization approach of Rockafellar and Uryasev is integrated into an IT architecture that is part of a sophisticated investment process. However, the approach of Rockafellar and Uryasev allows the implementation of additional constraints (e.g., transaction costs, liquidity constraints) which could considered in a more enhanced implementation as the one proposed in the present thesis. Additionally, further studies could compare the performance of the Rockafellar and Uryasev approach to the mean-variance optimization with a monthly re-balancing of a portfolio over the past decade where the performances of the two optimization approaches, during recent financial crisis, is of major concern.

Appendix A

Matlab Code

The implementation of the Rockafellar and Uryasev optimization approach and the corresponding analysis for the current thesis is done with Matlab R2014a. The code is implemented with functions and scripts. The functions accept input arguments and produce output. They are called from several scripts and are used to reduce code duplications and increase usability.

A.1 Functions

A.1.1 Optimization Algorithm

```
function [w, fval, exitflag, output, history] = optcvarmaxr(r, beta, CVaRLimit,
UB, LB, showPath, showIter)

% Sizes
[nSims, nAssets] = size(r);

% Inequality constraints
A1 = [zeros(1, nAssets) 1 1/(1-beta)*1/nSims*ones(1,nSims)];
A2 = -r;
A3 = -ones(nSims, 1);
A4 = -eye(nSims, nSims);

A = [A2 A3 A4];
A = [A1; A];

b = [-CVaRLimit zeros(1, nSims)];
b = b';

% Equality constraints --> sum of weights has to be 100%
Aeq = [ones(1, nAssets) zeros(1, nSims+1)];
beq = [1];
```

```
% Upper and lower bounds
UB = [repmat(UB, 1, nAssets) +Inf*ones(1, nSims+1)];
LB = [repmat(LB, 1, nAssets) zeros(1, nSims+1)];

% Initial weights and initial VaR
w0 = [(1/nAssets)*ones(1, nAssets)];
VaR0 = quantile(r*w0', beta);
w0 = [w0 VaR0 zeros(1, nSims)];

% Objective function
objfun = @(w) -mean(r(:,1:nAssets))*w(1:nAssets)';

% Optimization
options = optimoptions(@fmincon,'Algorithm','interior-point');
options = optimoptions(options,'MaxFunEvals',100000);
if(showIter)
options = optimoptions(options,'Display','iter');
end

if(showPath)
[w, fval, exitflag, output, history] = fminconpath(objfun, w0, A, b, Aeq, beq, LB, UB, options);
else
[w, fval, exitflag, output] = fmincon(objfun, w0, A, b, Aeq, beq, LB, UB, [], options);
history = [];
end
end
```

A.1.2 Optimization Path

```
function [w, fval, exitflag, output, history] = fminconpath(objfun, w0, A, b, Aeq, beq,
LB, UB, options)

history.x = [];
history.fval = [];

options = optimoptions(options,'OutputFcn',@outfun);
[w, fval, exitflag, output] = fmincon(objfun, w0, A, b, Aeq, beq, LB, UB, [], options);

function stop = outfun(x, optimValues, state)
     stop = false;

     switch state
         case 'init'
         case 'iter'
            % Concatenate current point and function value with history
            history.fval = [history.fval; optimValues.fval];
            history.x = [history.x; x];
         case 'done'
         otherwise
     end
 end

end
```

A.1.3 Correlated Asset Paths

```
function [S] = gbmcorr(S0, mu, sig, corr, dt, nSteps, nSims)
    nAssets = length(S0);
    drift = mu - sig.^2/2;
    R = chol(corr);
    S = nan(nSteps+1,nAssets,nSims);

    for idx = 1:nSims
        W = randn(nSteps, nAssets);
        corrW = W*R;
        S(:,:,idx) = [ones(1,nAssets);cumprod(exp(repmat(drift*dt,nSteps,1)
                                +corrW*diag(sig)*sqrt(dt)))]*diag(S0);
    end
end
```

A.1.4 Boostrapping

```
function [periodReturns] = bootstrap(R, nDays, nSims)
    nAssets = size(R,2);
    periodReturns = nan(nSims, nAssets);
    for iSim = 1:nSims
        dReturns = [ones(1, nAssets); nan(nDays, nAssets)];
        for day = 1:nDays
            pos = randi(length(R));
            dReturns(day+1, :) = R(pos, :)+1;
        end
        prices = cumprod(dReturns);
        periodReturns(iSim, :) = (prices(end,:) - prices(1,:))./prices(1,:);
    end
end
```

A.1.5 Weight Combinations

```
function W = weights(S0, nWeightCombis)
    nAssets = size(S0, 2);
    W=rand(nWeightCombis,nAssets);
    rowsums=repmat(sum(W,2),1,size(W,2));
    W=W./rowsums;
end
```

A.1.6 VaR

```
function [VaR] = varconf(r, confLvl)
    VaR = quantile(r, 1 - confLvl); % e.g. 1 - 0.9 = 0.1
end
```

A.1.7 CVaR

```
function [CVaR] = cvar(r, VaR)
    nAssets = size(r, 2);
    CVaR = nan(1, nAssets);
    for iAsset = 1:nAssets
        rWi = r(:,iAsset);
        CVaR(iAsset) = mean(rWi(rWi < VaR(iAsset)));
    end
end
```

A.1.8 Returns of Simulated Prices

```
function [r] = sims2ret(S)
    nAssets = size(S, 2);
    nSims = size(S, 3);

    r = nan(nSims, nAssets);
    for iSim = 1:nSims
        paths = squeeze(S(:,:,iSim));
        rSim = (paths(end,:) - paths(1,:))./paths(1,:);
        r(iSim,:) = rSim;
    end
end
```

A.1.9 Prices to Returns

```
function r = price2sret(p)
    r = zeros(size(p,1)-1, size(p,2));
    for i = 2:size(p,1)
        r(i-1, :) = (p(i, :) - p(i-1, :)) ./ p(i-1, :);
    end
end
```

A.1.10 Normalize Prices

```
function [dates, prices] = normPrices(marketdata, normPrice)
    dates = marketdata(:, 1);
    nTradingDays = size(marketdata, 1);
    startPrice = marketdata(round(nTradingDays/2), 2:end);
    prices = [];
    for cIdx = 1:7
        prices(:, cIdx) = marketdata(:, cIdx+1) ./ startPrice(1, cIdx) * normPrice;
    end
end
```

A.1.11 Plot Histogram

```
function [] = plothist(fig, data, t, xl, yl)
    figure(fig)
    x = data;

    [hts,ctrs] = hist(x,100);
    bh = bar(ctrs,hts,'hist');
    set(bh, 'facecolor', [0.5 0.5 1]);
    area = sum(hts) * (ctrs(2)-ctrs(1));
    xx = linspace(min(x), max(x));
    hold on;

    plot(xx,area*normpdf(xx,mean(x),std(x)),'b', 'Linewidth', 2)
    f = ksdensity(x,xx);
    plot(xx,area*f,'r-', 'Linewidth', 2)
    legend('Histogram', 'Normal Distribution', 'Approximated Distribution')
    title(t);
    xlabel(xl);
    ylabel(yl);
    grid on;
    hold off
```

A.1.12 Plot CCDF

```
function [] = ccdf(fig, data, data2, xl, yl, t)
    figure(fig)
    [f, xx] = ecdf(data);
    plot(xx, 1-f, 'red')
    hold on;

    if (isempty(data2) == 0)
        [f2, xx2] = ecdf(data2);
        plot(xx2, 1-f2, 'blue')
    end

    xlabel(xl);
    ylabel(yl);
    title(t);
    grid on;
    hold off;
end
```

A.1.13 Portfolios for Weight Combinations

```
function [P] = portfolios(W, r)
    P = r * W';
end
```

A.2 Scripts

A.2.1 Visualize Historical Prices

```
Prices = xlsread('Prices_126.xlsx');
[dates, prices] = normPrices(Prices, 100);
prices;

figure(11);
plot(dates, prices, 'LineWidth', 2)
grid on;
datetick('x','dd.mm.yy');
ylabel('Price [%]');
title('Asset Prices');
axis tight;
legend('CASH', 'CORB', 'GOVB', 'GOLD', 'COMM', 'MSCI', 'SP500');
hold off;
```

A.2.2 Monte Carlo Simulation

```
% Load prices
if(isempty(Prices))
    Prices = xlsread('Prices_126.xlsx');
end

% Initialize model
tDaysY = 252;
lookback = 126; % 252 trading days / 2 = 126 --> 126 / 7 = 18
p0 = Prices(end-lookback:end, 2:end);
r0 = price2sret(p0);   % get daily returns from prices

% Generate random correlated paths
S0 = ones(1, size(Prices,2)-1);
nSims = 1000;   % number of simulations
dt = 1/tDaysY;  % time steps --> days
nSteps = 10;    % days to expiry
S = gbmcorrhist(S0, r0, dt, nSteps, nSims);

% Calculate total returns from generated asset paths for the specific period
r = sims2ret(S);
corr_r = corrcoef(r)
```

A.2.3 Bootstrapping

```
% Load prices
if(isempty(Prices))
    Prices = xlsread('Prices_126.xlsx');
end

% Initialize model
tDaysY = 252;
lookback = tDaysY/2; % 252 trading days / 2 = 126 --> 126 / 7 = 18
p0 = Prices(end-lookback:end, 2:end);
r0 = price2sret(p0);   % get daily returns from prices

S0 = ones(1, size(Prices,2)-1);
nSims = 1000;           % number of simulations
days = 10;              % days to expiry
r = bootstrap(r0, days, nSims);
corr_r = corrcoef(r)
```

A.2.4 Optimization with CVaR Constraints

```
% Optimize portfolio weights for maximum return for a given risk limit
beta = 0.95; % Confidence level ?%
UB = 1.00;
LB = 0.00;

CVaRLimit = -2.50 % -2.5% loss in 10 trading days with confidence level = 95 %
%CVaRLimit = -2.00 % -2.0% loss in 10 trading days with confidence level = 95 %
%CVaRLimit = -1.50 % -1.5% loss in 10 trading days with confidence level = 95 %
%CVaRLimit = -1.00 % -1.0% loss in 10 trading days with confidence level = 95 %
%CVaRLimit = -0.50 % -0.5% loss in 10 trading days with confidence level = 95 %

CVaRLimit = CVaRLimit / 100;

showPath = true % true = show optimization path
showIter = true; % true = show optimization iterations
[w, fval, exitflag, output, history] = optcvarmaxr(r, beta, CVaRLimit, UB, LB, showPath, showIter);

% Display solver output
exitflag
output


% Values at optimal cvar-portfolio
nAssets = size(r, 2);

wOpt = w(1:nAssets)';
wOpt_p = wOpt * 100

R_Opt = -fval;
VaR_rOpt = -w(nAssets+1);

% Equally Weighted Portfolio
```

```
wEq = [(1/nAssets)*ones(1, nAssets)]';
rEq = r * wEq;
mu_rEq = mean(rEq)
sig_rEq = std(rEq);
VaR_rEq = varconf(rEq, 0.95);
CVaR_rEq = cvar(rEq, VaR_rEq)

% Optimal Portfolio
rOpt = r * wOpt; % daily return vector of portfolio

sig_rOpt = std(rOpt);
CVaR_rOpt = cvar(rOpt, VaR_rOpt)

if(isempty(CvarVsReturnOpt))
    CvarVsReturnOpt = [CVaR_rOpt_p R_Opt_p];
    CvarVsReturnEq = [CVaR_rEq_p mu_rEq_p];
else
    CvarVsReturnOpt = [CvarVsReturnOpt; CVaR_rOpt_p R_Opt_p];
    CvarVsReturnEq = [CvarVsReturnEq; CVaR_rEq_p mu_rEq_p];
end


% Display search path
if (isempty(history) == 0)
    weightsHistory = history.x(:, 1:nAssets) * 100;
    plotdata(2, weightsHistory, 'area',
                'Optimization Iterations',
                'Asset Weights [%]', 'Search Paths for Optimal Portfolio Weights');
    rHistory = -history.fval * (252/10) * 100; % annualize expected return history
    plotdata(4, rHistory, 'plot',
                'Optimization Iterations',
                'Return [%]', 'Search Path for Optimal Portfolio Return');
    plothist(6, rOpt * 100,
                'Distribution of the 10-day Returns of the Optimized Portfolio',
                'Return [%]', 'Frequency (out of 1000)');

    figure(21)
    [f, xi] = ksdensity(rOpt * 100);
    plot(xi, f, 'b', 'Linewidth', 2)
    axis([-4, 4, 0, 1.4]);
    title('Distribution of the 10-day Returns of the Optimized Portfolio');
    xlabel('Return [%]');
    ylabel('Density');
    legend('CVaR Limit = -2.5%', 'CVaR Limit = -2.0%',
                'CVaR Limit = -1.5%', 'CVaR Limit = -1.0%', 'CVaR Limit = -0.5%');
    grid on;
    hold on;
end
```

A.2.5 Brute-Force Analysis

```
% Generate weight combinations
W = weights(S0, 1000);

% Generate portfolios with different weight combinations
P = portfolios(W, r);
sizeR = size(r)
sizeP = size(P)

% Measure
mu = mean(P);
sig = std(P);
VaR = varconf(P, 0.95);
CVaR = cvar(P, VaR);

% Values at optimal CVaR Portfolio
maxCVaR = max(CVaR)
iMaxMuCVaRLevel = find(CVaR == maxCVaR);
maxCVaR_VaR = VaR(iMaxMuCVaRLevel)
maxCVaR_mu = mu(iMaxMuCVaRLevel)
maxCVaR_W = W(iMaxMuCVaRLevel, :)
maxCVaR_r = P(:, iMaxMuCVaRLevel);
maxCVaR_sig = sig(iMaxMuCVaRLevel);

% Values at optimal VaR Portfolio
maxVaR = max(VaR)
iMaxVaR = find(maxVaR == VaR);
maxVaR_mu = mu(iMaxVaR)
maxVaR_W = W(iMaxVaR, :)

% Values at optimal Variance Portfolio
minSig = min(sig)
iMinSig = find(sig == minSig);
minSig_mu = mu(iMinSig)
minSig_W = W(iMinSig, :)
minSig_r = P(:, iMinSig);

% Search Portfolio with Max Return and a certain CVaR limit
mu = mu * 100;
CVaR = CVaR * 100;

level = -2.50; % percent
[maxMuCVaRLevel_1, cvarMaxMu_1, sigMaxMu_1, rMaxMu_1]
= portcvarlevel(level, CVaR, sig, mu, P);

level = -2.00; % percent
[maxMuCVaRLevel_2, cvarMaxMu_2, sigMaxMu_2, rMaxMu_2]
= portcvarlevel(level, CVaR, sig, mu, P);

level = -1.50; % percent
[maxMuCVaRLevel_3, cvarMaxMu_3, sigMaxMu_3, rMaxMu_3]
= portcvarlevel(level, CVaR, sig, mu, P);

level = -1.00; % percent
```

```
[maxMuCVaRLevel_4, cvarMaxMu_4, sigMaxMu_4, rMaxMu_4]
= portcvarlevel(level, CVaR, sig, mu, P);

level = -0.50; % percent
[maxMuCVaRLevel_5, cvarMaxMu_5, sigMaxMu_5, rMaxMu_5]
= portcvarlevel(level, CVaR, sig, mu, P);

% Annualize Mu
mu = mu * (252 / 10);
plotdataxy(8, CVaR, mu, 'CVaR [%]',
'Return [%]', 'Generated Portfolios with Brute-Force Method',
'bx', cvarMaxMu_1, maxMuCVaRLevel_1, cvarMaxMu_2, maxMuCVaRLevel_2,

% Max Return
maxMu = max(mu)
iMaxR = find(mu == maxMu)
maxR_r = P(:, iMaxR);

plothist(10, rMaxMu_5 * 100,
'Distribution of Daily Returns of Optimal Portfolio',
'Return [%]', 'Frequency (out of 1000)');

figure(31)
[f, xi] = ksdensity(rMaxMu_5 * 100);
plot(xi, f, 'b', 'Linewidth', 2)
axis([-4, 4, 0, 2.0]);
title('Distribution of the 10-day Returns of the Optimized Portfolio');
xlabel('Return [%]');
ylabel('Density');
legend('CVaR Limit = -2.5%', 'CVaR Limit = -2.0%',
'CVaR Limit = -1.5%', 'CVaR Limit = -1.0%', 'CVaR Limit = -0.5%');
grid on;
hold on;
```

Bibliography

[1] Acerbi, C. and Tasche, D. On the coherence of expected shortfall. *Journal of Banking & Finance*, 26(7):1487–1503, 2002.

[2] Artzner, P., Delbaen, F., Eber, J.-M., and Heath, D. Thinking coherently. *Risk*, 10:68–71, 1997.

[3] Artzner, P., Delbaen, F., Eber, J.-M., and Heath, D. Coherent measures of risk. *Mathematical Finance*, 9(3):203–228, 1999.

[4] Brealey, R. A., Myers, S. C., and Franklin, A. *Principles of Corporate Finance*. McGraw-Hill, New York, 2012.

[5] Chen, P. and Xiong, J. Asset allocation in a non-normal world, 2009.

[6] Croarkin, C., Tobias, P., and Zey, C. *Engineering Statistics Handbook*. National Institute of Standards and Technology, 2001.

[7] Drake, P. P. and Fabozzi, F. J. *The Basics of Finance: An Introduction to Financial Markets, Business Finance, and Portfolio Management*. John Wiley & Sons, New Jersey, 2010.

[8] Einhorn, D. and Brown, A. Private profits and socialized risk. *Global Association of Risk Professionals*, 42:10–26, 2008.

[9] Elliott, R. J. and Kopp, P. E. *Mathematics of Financial Markets*, volume 10. Springer, New York, 2005.

[10] Elton, E. J., Gruber, M. J., Brown, S. J., and Goetzmann, W. N. *Modern Portfolio Theory and Investment Analysis*. John Wiley & Sons, New Jersey, 2009.

[11] Glasserman, P. *Monte Carlo Methods in Financial Engineering*. Springer, New York, 2004.

[12] Grüninger, S. Climbing returns and falling meteorites. *Credit Suisse Bulletin*, (2):27, 2013.

[13] Jorion, P. Bayesian and capm estimators of the means: Implications for portfolio selection. *Journal of Banking & Finance*, 15(3):717–727, 1991.

[14] Jorion, P. *Value at Risk: The New Benchmark for Managing Financial Risk*, volume 3. McGraw-Hill, New York, 2007.

[15] Krokhmal, P., Palmquist, J., and Uryasev, S. Portfolio optimization with conditional value-at-risk: Objective and constraints. *Journal of Risk*, 4:43–68, 2002.

[16] Markowitz, H. Portfolio selection. *Journal of Finance*, 7(1):77–91, 1952.

[17] Meindl, B. and Templ, M. Analysis of commercial and free and open source solvers for linear optimization problems. 2012.

[18] Nyholm, K. *Strategic Asset Allocation in Fixed Income Markets: A Matlab Based User's Guide*. John Wiley & Sons, New Jersey, 2008.

[19] Rachev, S. T., Menn, C., and Fabozzi, F. J. *Fat-Tailed and Skewed Asset Return Distributions: Implications for Risk Management, Portfolio Selection, and Option Pricing*. John Wiley & Sons, New Jersey, 2005.

[20] Rockafellar, R. T. and Uryasev, S. Optimization of conditional value-at-risk. *Journal of Risk*, 2:21–42, 2000.

[21] Ross, S. M. *Introduction to Probability Models*. Academic Press, Oxford, 10 edition, 2014.

[22] Sarykalin, S., Serraino, G., and Uryasev, S. Value-at-risk vs. conditional value-at-risk in risk management and optimization. *Tutorials in Operations Research*, 2008.

[23] Siegel, L. B. and Kaplan, P. D. *Frontiers of Modern Asset Allocation*. John Wiley & Sons, New Jersey, 2011.

[24] Stillhart, G. The same but different. *Credit Suisse One*, (3):24–26, 2012.

[25] Stillhart, G., Grüninger, S., Alex, P., and Gunevski, V. Asset allocation advisory for institutional and private clients, 2014.

[26] Uryasev, S. Conditional value-at-risk: Optimization algorithms and applications. *Financial Engineering News*, (14):49–57, 2000.

[27] Yang, W. Y., Cao, W., Chung, T.-S., and Morris, J. *Applied Numerical Methods Using Matlab*. John Wiley & Sons, New Jersey, 2005.