POLITECNICO DI TORINO

Master's Degree in Mathematical Engineering

Master's Degree Thesis

An application of moment matching and L-shaped decomposition to stochastic portfolio optimization

Supervisors

Prof. Edoardo FADDA

Prof. Paolo BRANDIMARTE

Candidate

Stefano PEZZUTO

October 2024

Summary

This thesis deals with stochastic portfolio optimization. In particular, we focus on two crucial aspects: The representation of uncertainty through the creation of a free-of-arbitrage scenario tree and the resolution of the portfolio optimization model by the L-shaped decomposition method. For the tree construction, we investigate and implemented a method called moment matching, comparing against the plain Monte-Carlo method that sample from Geometric Brownian Motion. Furthermore, we focus on how to exclude arbitrage opportunities during the generation of the scenario tree. For the resolution of the model, we use an L-shaped decomposition. All the methods presented have been implemented and tested on some instances, showing the effectiveness of both the moment matching method and the L-shaped decomposition.

A Sara, Kika, Carlo e Davide,
A mia mamma e mio padre,
Alla mia splendida Alice,
Grazie per la fantastica famiglia che siamo!

Table of Contents

List of Tables

List of Figures

Acronyms

MM
: moment matching

GBM
: Geometric Brownian Motion

LP
: Linear Programming

Chapter 1

Introduction

Portfolio optimization has long been a central topic in finance, focusing on the optimal allocation of assets. The pioneriing work of Harry Markowitz in the 1950s laid the foundation for the modern portfolio theory, based on maximizing returns and simultaneously minimizing risk [1]. Since that work, portfolio optimization has evolved significantly, largely due to advancements in Operations Research. Real-world investment decisions often involve uncertainty and require more sophisticated models that can capture the complexities of financial markets. This is where stochastic optimization plays a crucial role, offering tools to incorporate uncertainty into decision-making processes. Scenario generation is a critical component of stochastic optimization as it allows to model uncertain future outcomes, such as asset prices evolution. In this context, the generation of a reliable scenario tree and the subsequent resolution of the optimization model are two fundamental aspects of stochastic portfolio optimization, which will be the focus of this thesis. Chapter 2 provides the theoretical background related to the key concepts of portfolio optimization and stochastic programming. Chapter 3 delves into the crucial step of scenario generation, where two different approaches are introduced: moment matching (MM) and Monte Carlo sampling from a Geometric Brownian Motion (GBM) process. These methods are used to simulate possible future states of the financial market, forming the basis for the subsequent optimization. The portfolio optimization model considered is presented in Chapter 4 along with how the L-Shaped decomposition approach works, including its extension to multistage problems using the nested L-Shaped method. Results of computational tests are illustrated and examined in Chapter 5. Finally, Chapter 6 concludes the thesis, summarizing the main findings. Through this structure, this thesis aims to provide a comprehensive approach to stochastic portfolio optimization, addressing scenario generation, model drafting and model resolution.

Chapter 2

Theoretical Background

2.1 Financial markets

Financial markets allow investors to buy and sell assets with two main purposes: Shifting consumption over time and transferring risk. An asset is any resource that holds value and can be converted into money by its owner. Securities are tradable (purchasable or saleable) financial asset. This thesis is focused on the share market. Stock shares are securities and they are inherently risky assets, as their future values cannot be predicted with certainty. However, risk-free assets also exist and provide a guaranteed future income. An example of a risk-free asset is a bank account at a large bank into which interest is paid. A portfolio is a collection of assets that represents the investor position in each asset. Considering N different assets, a portfolio can be represented as a vector

$$\boldsymbol{x} = (x_1, x_2, ..., x_N, x_{N+1}),$$

where, x_i for $i = 1, ..., N$, represents the investor position in asset i and x_{N+1} stands for the liquidity in the bank account, which is considered as the risk-free asset and guarantees a periodical income based on the risk-free rate of the market and the amount of money in the bank account. Specifically, for the remainder of this project, x_i for $i = 1, ..., N$ is interpreted as the number of shares of asset i owned by the investor, but in general it could be also often interpreted as the fraction of the total portfolio value that the investment in asset i represents. Given asset prices vector: $\mathbf{p} = (p_1, ..., p_N)$, it is possible to determine the portfolio value (in dollars or euros), which is simply given by the market value of the owned shares plus the liquidity in the bank account, namely

$$V(\mathbf{x}) = \left(\sum_{i=1}^{N} p_i x_i \right) + x_{N+1}.$$

2.1.1 Asset prices dynamic

The price of an asset i represents its market value and it could be modeled through a discrete time series $p_i^{[t]}$. To measure the profitability of an asset and the variation in its price the concept of return is usually employed. For an asset i, the *linear return* over an holding period $[0, t_2]$ is defined as

$$r_i^{[t_2]} = \frac{p_i^{[t_2]} - p_i^{[0]}}{p_i^{[0]}}. \tag{2.1}$$

The biggest drawback of linear returns is that they are not additive, which means that, considering $0 \leq t_1 \leq t_2$, it is not possible to compute the overall linear return over the period $[0; t_2]$ just by summing the linear return in $[0; t_1]$ and the one in $[t_1; t_2]$. For this reason, *log-returns* are very often used to model asset prices variation. The log-return of an asset i over an holding period $[0, t_2]$ is defined as

$$R_i^{[t_2]} = \log\left(\frac{p_i^{[t_2]}}{p_i^{[0]}}\right). \tag{2.2}$$

Log-returns are additive, in fact it holds that

$$R_i^{[t_2]} = \log\left(\frac{p_i^{[t_2]}}{p_i^{[0]}} \cdot \frac{p_i^{[t_1]}}{p_i^{[t_1]}}\right) = \log\left(\frac{p_i^{[t_2]}}{p_i^{[t_1]}}\right) + \log\left(\frac{p_i^{[t_1]}}{p_i^{[0]}}\right). \tag{2.3}$$

2.2 Stochastic multistage programming

Stochastic programming is a mathematical framework for modeling optimization problems that involve decision-making under uncertainty. Unlike deterministic models, which assume that all problem parameters are known, stochastic optimization takes into account randomness and unpredictability while optimizing criteria chosen by the decision maker. There exist two different approaches to tackle stochastic problems. If only here-and-now decision are made and thereafter results are observed after a given time period, then a static decision problem is being considered, since the possibility of adjusting the decisions along the way is disregarded, not taking into consideration the actual unfolding of uncertain risk factors. On the other side, in multistage decision models the possibility of updating decisions, depending on the incoming information flow over time, is taken into account. It is worth emphasizing the distinction between stages and periods. A period is a future time instant in the considered time period in which realizations of involved random variables are observed. A stage is a future time instant in the considered time period in which the decision maker may make some actions

based on the information acquired in time. Hence, a multiperiod problem requires the planning of decisions to be executed over a sequence of time instants, but the problem is actually static if these decisions are all taken at the initial instant (here-and-now). On the contrary, in a multistage setting, the solution consists of a set of actions that are contingent upon the realization of random factors, so the decision maker perform actions according to the observed state of the uncertain world at each stage.

2.2.1 Stochastic multistage portfolio optimization

Portfolio optimization involves optimizing investment strategies selecting the composition of the portfolio, namely the shares number to own for each asset, to achieve specific financial objectives. In this context, stochastic programming allows to take into consideration uncertainty in future asset prices, which are obviously unknown, and multistage stochastic optimization allows to rebalance the portfolio in fixed future instants based decisions on new information and new market conditions. In this dynamic setting, the portfolio value changes over time according to prices evolution and the rebalancing of positions performed by the investor. In particular, the portfolio composition at a given time t is

$$\boldsymbol{x}^{[t]} = \left(x_1^{[t]}, x_2^{[t]}, ..., x_N^{[t]}, x_{N+1}^{[t]}\right),$$

and its value is given by

$$V\left(\boldsymbol{x}^{[t]}\right) = \left(\sum_{i=1}^{N} p_i^{[t]} x_i^{[t]}\right) + x_{N+1}^{[t]}.$$

To sum it up, in multistage stochastic portfolio optimization, the investment horizon is divided into several stages; at each stage, investors make decisions based on the observed current state of the market and projections for future stages. This approach allows for dynamic adjustments to the portfolio, aiming to optimize the expected returns while managing risks throughout the investment period in a sense that will be better explained in future sections.

2.3 Risk aversion and utility functions

In finance it is often assumed that investors are risk-averse. A decision maker is said to be risk-averse if when she is in the position of choosing between two lotteries X and Y with the same payoff but different degree of risk, she will choose the one with lower risk. For example, let consider a lottery X that guarantees a sure payoff μ and a lottery Y which offers two equally likely payoffs $\mu - \sigma$ and $\mu + \sigma$, then

a risk-averse decision maker will arguably prefer the lottery X. In the context of portfolio optimization, one may think that the unique goal of an investor is to maximize the expected final wealth, but this is not necessarily true, in fact usually this does not reflect investor behavior. To understand this concept it is good to present the Saint Petersburg paradox. Let imagine that a lottery is offered to a decision maker, the outcome depends on the flip of a fair and memoryless coin that is flipped until it lands tails. The payoff is 2^k, where k is the number of times that the coin lands heads. How much should the decision maker be willing to pay for this lottery? In other words, what is the fair price of this lottery? One may think that the fair price is the expected payoff. Since events are independents the probability of winning 2^k is $\frac{1}{2^{k+1}}$, then the expected value of the payoff is

$$\sum_{k=0}^{\infty} \frac{1}{2^{k+1}} 2^k = +\infty \tag{2.4}$$

but nobody would probably pay an unlimited amount of money to play this lottery, that surely offers huge payoffs but with vanishing probabilities [2]. Hence, in order to appropriately model the investor behavior the expected final wealth is not the only factor to be evaluated and risk-aversion should be taken into consideration. One way to express risk-aversion is based on the assumption that the decision maker put random variables in order through a functional called *Von Neumann-Morgenstenn expected utility*, defined as

$$U(X) = \mathbb{E}\left[u(X)\right], \tag{2.5}$$

where X is a random variable representing a payoff and $U(\cdot)$ is a functional since it maps random variables to real numbers allowing to establish an order among random variables. The function $u(\cdot)$ is called utility function and requires certain characteristics depending on the application field. In finance, it appears reasonable to require the utility function $u(\cdot)$ to be strictly increasing, as greater wealth typically leads to greater satisfaction for the investor. This concept is known as *non-satiation property*. Secondly, an utility function should be concave to express risk-aversion, meaning the investor's preference for a certain amount rather than a bet with the same expected value. This is due to the Jensen's inequality, which states that for a concave function u and a random variable X, it holds that

$$u\left(\mathbb{E}\left[X\right]\right) \geq \mathbb{E}\left[u\left(X\right)\right]. \tag{2.6}$$

To better illustrate the concept, consider again a lottery X which guarantees a sure payoff μ and a lottery Y which offers two equally likely payoffs $\mu - \sigma$ and $\mu + \sigma$ (Y is a mean-preserving transformation of X). Then,

$$U\left(X\right) = u(\mu) = u\left(\mathbb{E}\left[Y\right]\right) \tag{2.7}$$

$$U\left(Y\right) = \frac{1}{2}u(\mu - \sigma) + \frac{1}{2}u(\mu + \sigma) = \mathbb{E}\left[u\left(Y\right)\right]. \tag{2.8}$$

For Jensen's inequality it holds that $U(X) \geq U(Y)$, which in fact reflect the investor's preference for a certain payoff rather than a bet with the same expected payoff [2].

Chapter 3

Scenario generation

3.1 Scenario Tree

A crucial point in optimization models under uncertainty is actually the representation of uncertainty itself. In the real world, it is plausible to think that, in many contexts, stochastic quantities (such as stock prices) follow a continuous probability distribution. However, as pointed out in [3], except for trivial cases, a discrete distribution is needed to effectively solve multistage stochastic problems. For this reason, a discrete approximation of the real continuous distribution is often employed. Scenario trees serve specifically as an approximate discrete representation of the problem's randomness: Each layer of the tree represents a different stage of the problem and each node of the tree represents a possible realization of uncertainty parameters (hence asset prices in the portfolio optimization context). As already pointed out in Section 2.2, periods and stages are not the same thing: a period is an instant in which the state of the world is observed, instead, a stage is an instant at which the decision maker can perform an action according to its strategy to optimize the objective function. However, in financial context (and not only), periods and stages often overlap, namely, moments when variables are observed and actions are performed are the same. Therefore, the terms period and stage will be used with the same meaning in the remainder of this thesis referring to a future instant of the scenario tree. Moreover, the set of nodes is called $\mathcal{N} = \{0,1,\dots,N\}$ and $\mathcal{N}^+ = \mathcal{N} \setminus \{0\}$. Node 0 is the root node, representing the present, where here-and-now decisions are made. At each stage, asset prices are represented by a node n with probability $\pi^{[n]}$ (for the root node it holds that $\pi^{[0]} = 1$) and actions could be performed to rebalance the portfolio according to the new observed realization of uncertainty. The parent of a node $n \in \mathcal{N}^+$ is denoted with $p(n)$. The set $\mathcal{N}_t \subset \mathcal{N}$ is the set of nodes at time t. As can be imagined, in general the larger the tree size the better the approximation of the

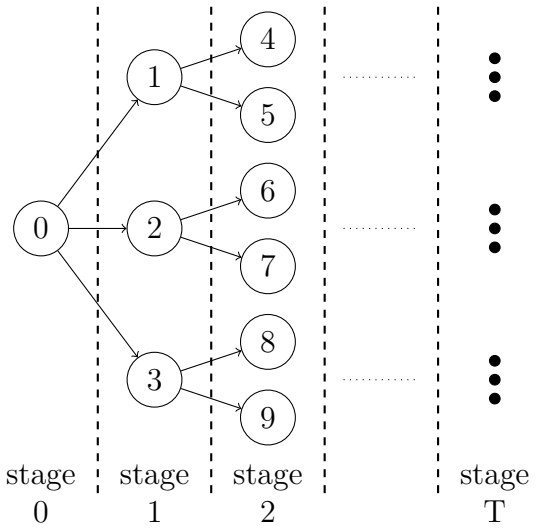

Figure 3.1: Example of a scenario tree representation.

true parameters distribution. In fact, a larger number of nodes would allow for greater accuracy and completeness in representing the different possible states of the system. Nevertheless, large trees require more computational effort, both during the generation and later when employed by the optimization algorithm. Therefore, it is essential to find a good trade-off, looking for a tree size that guarantees both a good approximation and that curse of dimensionality does not arise. In portfolio optimization, the scenario tree has to be generated to model the potential future evolution of the stock market, where each node in the tree represents a possible future state of stocks' prices. At each of these nodes, the decision maker is presented with the opportunity to execute actions like buying and/or selling assets. These actions are critical as they allow the decision maker to adjust the portfolio in response to the changing market conditions, aiming to optimize the overall performance of the portfolio. The generation of the scenario tree is a crucial first step in obtaining a successful portfolio optimization algorithm. For this reason, numerous methods have been studied in literature to generate the scenario tree appropriately. Section 3.2 discuss about arbitrage opportunities in the market and how they affect portfolio optimization problem. Section 3.3 shows the general structure of the code implemented. Then, Sections 3.4 and 3.5 present techniques used to build the scenario tree in this project.

3.2 Arbitrage opportunities

An arbitrage is a trading strategy that ensures a risk-free profit without the need for initial capital. In other words, an arbitrage is a strategy that creates money from nothing [4]. Almost all financial models assume the absence of arbitrage opportunities in the market, mainly for two reasons. Firstly, although such opportunities may arise in reality, the market will soon eliminate them, thereby establishing a new equilibrium that does not contemplate arbitrage. This happens because in financial markets there are many people, called arbitrageurs, that look for arbitrage opportunities to exploit them [5]. Furthermore, as stated in [4] and in [5], if the scenario tree admits arbitrage opportunities, then the portfolio optimization solution will be biased. In particular, if the investor can exploit arbitrage opportunities, the optimal solution will be unbounded. Yet, even if there were no possibility of exploiting arbitrage opportunities due to market imperfections (frictions) or restrictions, the presence of such opportunities may cause substantial biases in the optimal investment strategy. For these reasons, arbitrage opportunities are usually excluded in modern asset pricing models and it appears reasonable to rule out them also while generating the scenario tree for the purpose of this thesis. Some notation is now introduced to formalize what an arbitrage is and how to exclude it while generating the scenario tree:

- M is the number of assets, indexed by i;
- $\mathcal{S}$ is the set of scenarios, indexed by s;
- $\boldsymbol{p}^{[0]} \in \mathbb{R}^M$ is the vector of initial asset prices (at $t = 0$);
- $P \in \mathbb{R}^{M \times S}$ is the matrix of asset prices in each scenario after one step, in particular each entry $P_{i,s}$ is the price of asset i after one time step in scenario s;
- $\mathbf{h} \in \mathbb{R}^M$ is a given trading strategy (a portfolio), where each h_i is the number of shares of the asset i owned by the investor.

As pointed out in [6] there are two types of arbitrage. An arbitrage opportunity of the first type can be mathematically formalized as

$$\sum_{i=1}^{M} h_i p_i^{[0]} = 0 \tag{3.1}$$

$$\sum_{i=1}^{M} h_i P_{i,s} \geq 0 \qquad \forall s \in \mathcal{S} \tag{3.2}$$

$$\exists s \in \mathcal{S} : \sum_{i=1}^{M} h_i P_{i,s} > 0. \tag{3.3}$$

In words, an arbitrage opportunity of the first type arises when there exists a trading strategy $\boldsymbol{h}$ such that the initial value of the portfolio is zero (3.1) and the portfolio's value will be non-negative in all future scenarios (3.2) and positive in at least one scenario (3.3). As a result, the expected payoff is strictly positive. To detect whether such opportunities are present, the following linear program can be solved

$$\max_{\boldsymbol{h}} \quad \mathbf{1}^\top \left(P^\top \boldsymbol{h}\right) \tag{3.4}$$

$$\text{s.t.} \quad \boldsymbol{p}^{[0]\top} \boldsymbol{h} = 0 \qquad \nu \tag{3.5}$$

$$P^\top \boldsymbol{h} \geq 0 \qquad \boldsymbol{\lambda}. \tag{3.6}$$

If a strictly positive solution exists for this linear problem then it indicates the presence of an arbitrage opportunity. In fact, constraints (3.5) and (3.6) ensure Equations (3.1) and (3.2) respectively, and the positivity of the solution guarantees that there is at least one scenario where the strategy $\mathbf{h}$ yields a positive return. Furthermore, if a solution exists, then the linear program will be unbounded, as the asset allocation $\mathbf{h}$ can be multiplied by any arbitrary constant without violating the constraints. According to the duality theory, if the primal problem is unbounded, the dual problem must be infeasible. Considering the dual variables $\nu \in \mathbb{R}$ and $\boldsymbol{\lambda} \in \mathbb{R}^S$, the dual of the problem above is

$$\min_{\nu, \boldsymbol{\lambda}} \quad 0 \tag{3.7}$$

$$\text{s.t.} \quad \boldsymbol{p}^{[0]}\nu - P\boldsymbol{\lambda} = P\mathbf{1} \tag{3.8}$$

$$\boldsymbol{\lambda} \geq 0. \tag{3.9}$$

Given that the objective function (3.7) is constant, determining the presence of arbitrage opportunities of the first type involves just verifying whether the linear system of equations specified by Equation (3.8) has a non-negative solution (as requested by constraint (3.9)). If such a solution exists, then no arbitrage opportunities of the first type are present.
An arbitrage opportunity of the second type may be formalized as

$$\sum_{i=1}^{M} h_i p_i^{[0]} < 0 \tag{3.10}$$

$$\sum_{i=1}^{M} h_i P_{i,s} \geq 0 \qquad \forall s \in \mathcal{S}. \tag{3.11}$$

Where Equation (3.10) states that the portfolio $\boldsymbol{h}$ has a negative initial value and Equation (3.11) guarantees that the strategy generate a non-negative payoff in every future scenario (since its value is greater or equal than 0 in every scenario).

Hence, to check whether arbitrage opportunities of the second type are present in the market

$$\min_{\boldsymbol{h}} \quad \boldsymbol{p}^{[0]\top}\boldsymbol{h} \tag{3.12}$$

$$\text{s.t.} \quad P^{\top}\boldsymbol{h} \geq 0 \qquad\qquad \boldsymbol{\gamma} \tag{3.13}$$

is the linear programming problem that should be solved. If there exists a trading strategy $\boldsymbol{h}$ such that this optimization problem has a negative optimum, then an arbitrage opportunity of the second type exists. Moreover, in such a case, the linear program will be unbounded since one can multiply the asset allocation $\boldsymbol{h}$ for an arbitrary constant without violating constraints. Introducing the dual variable $\boldsymbol{\gamma} \in \mathbb{R}^S$, by duality theory, it would mean that the dual of the above problem

$$\max_{\boldsymbol{\gamma}} \quad 0 \tag{3.14}$$

$$\text{s.t.} \quad P\boldsymbol{\gamma} = \boldsymbol{p}^{[0]} \tag{3.15}$$

$$\boldsymbol{\gamma} \geq 0 \tag{3.16}$$

is unfeasible. As in the previous case, to assess whether there is an arbitrage opportunity of the second type in the market, the system of equations defined by constraint (3.15) has to be solved and if there is a non-negative solution, then arbitrage opportunities of the second type do not arise. Focusing on the equations system (3.15), it is straightforward to write it like

$$\sum_{s\in\mathcal{S}} \boldsymbol{\gamma}_s P_{i,s} = \boldsymbol{p}_i^{[0]} \qquad\qquad \forall i \in \{1, ..., M\}, \tag{3.17}$$

in this way it is easier to see that the equation system states that the market is free of arbitrage if there exists a probability measure that makes the expected future value of the asset equal to the current value (in fact if $\boldsymbol{\gamma}$ is rescaled in such a way that its entries sum up to one, it could be interpreted as a probability vector). This is the fundamental property of martingale measures.
For the reasons expressed at the beginning of the section, it's important to exclude arbitrage opportunities (of both first and second type) while generating the scenario tree that will be used by the stochastic multistage programming, otherwise the solution will be biased. Two key facts could help in in reducing computational effort in the generation of the tree. Firstly, if the set of Equations (3.15) has a strictly positive solution, then no arbitrage opportunities of either the first or the second type are present; secondly, in presence of a risk-free asset (bank account) in the market, the two arbitrage opportunities are equivalent to each other. Furthermore, as stated in [5] and in [7], to appropriately choose the tree size it is important to keep in mind that a necessary condition to obtain an arbitrage free scenario tree is that the branching factor (namely, the number of arcs emanating from a node) at each node of the tree must be at least equal to the number of non-redundant assets in the optimization problem.

3.3 Code Structure

The scenario tree is built using a well-defined code structure that leverages various classes and functions. The key components of this structure are as follows:

- Class `ScenarioTree`: This class utilizes the *networkx* library to build the scenario tree. It serves as the main framework for the tree construction process.
- Abstract Class `StochasticModel`: This abstract class is designed to be extended by specific stochastic models. It is called by the `ScenarioTree` class to iteratively add nodes to the tree based on a chosen model. Two concrete implementations of this class have been developed:
  - `GeometricBrownianMotion`
  - `MomentMatching`

  Each of these models will be detailed in the following paragraphs and provide a distinct approach for generating the scenario tree.
- Function `check_arbitrage`: This function evaluates the presence of arbitrage opportunities. It takes as input the current node and a set of candidate child nodes, checking for arbitrage as described in Section 3.2.

The process begins by selecting an instance of the `StochasticModel` class. For each node (excluding the leaves) in the tree, the `ScenarioTree` class invokes this instance, passing all necessary parameters to generate a specified number of child nodes. Once the matrix of log-returns, R_{t+1}, is generated, the corresponding asset prices matrix, P_{t+1}, is computed easily using the formula

$$P_{t+1}[i, s] = p_t[i] \cdot \exp\left(R_t[i, s]\right), \tag{3.18}$$

where $\boldsymbol{p}_t$ represents the current asset prices vector. After determining the possible future market states, the matrix P_{t+1} and the vector $\boldsymbol{p}_t$ are passed to the `check_arbitrage` function. If no arbitrage opportunities are found, the new nodes are added to the tree; otherwise, the process is repeated with a new set of candidates. More details about the code in Appendix A.

3.4 Moment matching

Moment matching (MM) is a method that was presented in [8] and it has received significant attention in literature because it has been shown to perform well in

the financial context [9]. The core idea of this method is to approximate the real distribution by generating the scenario tree pursuing the goal of minimizing the distance between some specified statistical properties and those of the discrete distribution underlying the tree. By aligning the moments of the discrete distribution with those of the real (probably continuous) distribution, the moment matching method ensures a more accurate and representative model of the possible future states of the market. Moreover, the number of nodes to generate is a parameter to be chosen, so the tree size can be kept under control.

3.4.1 Single-period tree

Since the generation of a multi-stage tree is nothing but the reiteration of multiple two-stage trees generation, it is useful to begin by introducing the model to generate a single-period tree with a defined number of scenarios. Let $\mathcal{V}$ be the set of statistical properties of interest and v_k is the specified value for the statistical property k. Let M be the number of assets in the market (indexed with $i \in \{1, ..., M\}$), S be the number of children to be generated (indexed with $s \in \{1, ..., S\}$); R is the matrix of assets log-returns, with dimension $M \times S$: each row corresponds to an asset, each column to a node, so each entry R_{is} is the log-return of the asset i in the scenario s of the tree. $\boldsymbol{\pi}$ is the probability vector, of dimension S, so that each π_s is the probability assigned to the scenario s. It is worth emphasizing that, generally speaking, the term scenario refers to an entire sample path of the tree, namely, from the start node to an ending node, but since in this section we are focusing on a single-period tree, each node at time $t = 1$ coincides with a scenario. Given returns R and probabilities $\boldsymbol{\pi}$ that arrange the tree, $f_k(R, \boldsymbol{\pi})$ is the function that computes the value that the statistical property k takes on the generated tree. The moment matching problem can be formalized as:

$$\min_{R,\boldsymbol{\pi}} \quad \sum_{k \in \mathcal{V}} (f_k(R, \boldsymbol{\pi}) - v_k)^2 \tag{3.19}$$

$$\text{s.t.} \quad \sum_{s=1}^{S} \pi_s = 1 \tag{3.20}$$

$$\pi_s \geq 0 \quad \forall s \in 1, ...S. \tag{3.21}$$

Log-returns matrix R and probabilities vector $\boldsymbol{\pi}$ are the decision variables that, once found, compose the tree. The objective function (3.19) is the sum of squared differences among the moments underlying the tree and the specified expected moments. Equations (3.20) and (3.21) imposes that probabilities sum up to 1 and should not be negative. It is important to stress few points. Firstly, this optimization problem is not convex in general, so appropriate methods must be chosen for resolution and a local optimum instead of a global one can be found.

Furthermore, it is not straightforward which statistical properties have to be specified and how many scenarios have to be generated, since these factors depends on many aspects: The portfolio optimization model that will be solved, the number of assets and the characteristics of the real distribution. However, the first 4 moments (mean, standard deviation, skewness and kurtosis) plus correlations are often used in literature for portfolio optimization problems and so they are used in this thesis as well. Table 3.1 contains formulas used to evaluate these moments for the generated tree. As authors pointed out in [8], number of scenarios that have to

Mean: $\boldsymbol{\mu} \in \mathbb{R}^M$	$\mu_i = \sum_{s=1}^{S} \boldsymbol{\pi}_s R_{is}$
Standard Deviation: $\boldsymbol{\sigma} \in \mathbb{R}^M$	$\boldsymbol{\sigma}_i = \sum_{s=1}^{S} \boldsymbol{\pi}_s \left(R_{is} - \boldsymbol{\mu}_i\right)^2$
Skewness: $\boldsymbol{t} \in \mathbb{R}^M$	$t_i = \sum_{s=1}^{S} \boldsymbol{\pi}_s \left(\frac{R_{is} - \boldsymbol{\mu}_i}{\boldsymbol{\sigma}_i}\right)^3$
Kurtosis: $\boldsymbol{\kappa} \in \mathbb{R}^M$	$l_i = \sum_{s=1}^{S} \boldsymbol{\pi}_s \left(\frac{R_{is} - \boldsymbol{\mu}_i}{\boldsymbol{\sigma}_i}\right)^4$
Correlation: $\rho \in \mathbb{R}^{M \times M}$	$\rho_{ij} = \sum_{s=1}^{S} \boldsymbol{\pi}_s \left(R_{is} - \boldsymbol{\mu}_i\right)\left(R_{js} - \boldsymbol{\mu}_j\right)$

Table 3.1: MM: How tree statistical moments have been calculated. R and $\boldsymbol{\pi}$ are log-returns and probabilities that compose the tree.

be generated and number of specified properties are related each other, and they recommend a very simple heuristic that can often lead to a minimal tree size that it is enough to closely have the match with the specified statistical properties. The heuristic is trivial and consists in selecting S (the number of scenarios) in such a way that the number of variables in the single-period tree is at least equal to the number of specifications. For example, if there are 5 assets and the first four moments plus correlations are specified, it means there are 30 specifications: $5 \cdot 4 = 20$ for the moments, plus 10 correlations between the assets. In this setting, the number of variables in the tree is given by $(5 + 1) \cdot S - 1$, since for each scenario there are 5 log-returns and 1 probability, but there is also the constraint for probabilities that reduce the degree of freedom (and that's why there is minus 1). It is straightforward that in this example the minimum number of scenarios that makes the number of variables greater or equal than 30 is $S = 6$. Moreover, a necessary condition to exclude arbitrage opportunities is that the number of children is greater or equal to the number of assets. In [6] it is stated that to rule out arbitrage opportunities one can verify whether the equation system (3.15) described in the Section 3.2

have non-negative solutions after the tree has been generated or, equally, put the same equations as constraints in the optimization model (3.19) - (3.21). The two approaches lead to the same result. However, placing those constraints within the optimization problem increases the non-convexity and the complexity of the problem itself, so in this project the arbitrage check is performed ex-post.

3.4.2 Multi-period tree

In a multistage setting, the model above should be generalized to achieve a representation of uncertainty over different periods. In particular, intertemporal dependencies should be taken into consideration to relate one period to the previous one. A good way to generate a tree with more than one period is simply to sequentially iterate the generation of a single period tree: Specify statistical properties for the first period and generate first-period outcomes that are consistent with these specifications; for each generated first-period outcome, specify conditional distribution properties for the second period, and generate conditional second-period outcomes that are consistent with these specifications; then continue to specify conditional distribution properties and generate consistent outcomes through all the periods. The expression *conditional distribution properties* means that intertemporal dependencies are expressed relating the specified properties used to generate the new single period scenario tree to the outcome of the previous period that is the root node of the subtree that is being created.

3.4.3 Implementation

To generate the tree using the moment matching method, the class `ScenarioTree` iteratively calls the instance `MomentMatching` of the class `StochModel` to generate a defined number of children from a parent node through the moment matching method, until the whole tree is created. The optimization problem (3.19) - (3.21) has been solved using the optimization module of `scipy` library. In particular, it has been solved using the `SLSQP` method of the library. The expected moments, fundamental parameters of the problem, were calculated statistically from historical data.

3.5 Geometric Brownian Motion

A simpler method to generate the scenario tree is by employing the Geometric Brownian Motion through Monte-Carlo sampling.

3.5.1 Wiener Process

To understand Geometric Brownian Motion, it's essential to first grasp the concept of Standard Brownian Motion (also known as Wiener Process), denoted by W_t, is a continuous-time stochastic process characterized by the following properties:

- Initial condition: $W_0 = 0$;
- Independent increments: $\forall\ 0 \leq t_0 < t_1 < \cdots < t_T$, $W_{t_1} - W_0, \ldots, W_{t_T} - W_{t_{T-1}}$ are independent random variables;
- Normal and stationary increments: $W_t - W_s$ is normally distributed with mean zero and variance $t - s$, $\forall t, s \in \{0, \ldots, t_T\}$;
- Continuous paths: the path of W_t is continuous, though not differentiable.

Standard Brownian Motion is often used to model the random fluctuations observed in various natural and financial phenomena due to its simplicity and well-defined statistical properties.

3.5.2 Geometric Brownian Motion

Geometric Brownian Motion extends the concept of Standard Brownian Motion to model the evolution of asset prices over time. It is particularly well-suited for financial modeling because it accounts for the empirical observation that asset prices tend to grow exponentially over time, rather than linearly. An asset price P_t

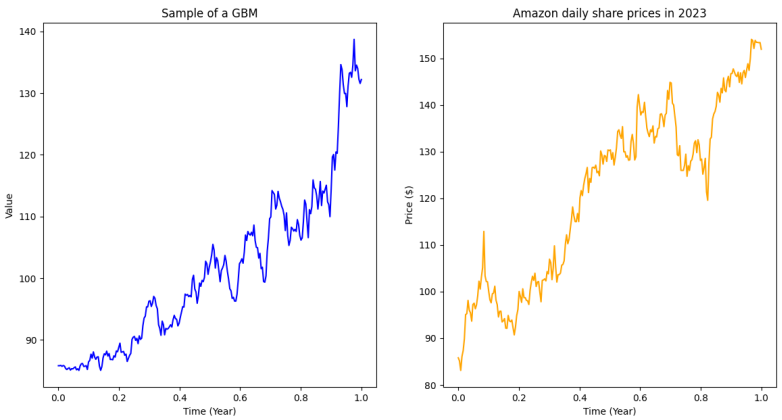

Figure 3.2: Simulation of a Geometric Brownian Motion process (with parameters estimated from historical data) compared to Amazon's daily stock prices in 2023

is said to follow Geometric Brownian Motion if it satisfies the following stochastic differential equation

$$dP_t = \mu P_t dt + \sigma P_t dW_t, \tag{3.22}$$

where:

- W_t is a Standard Brownian Motion;
- μ is called drift parameter, representing the average rate of asset returns;
- σ is called volatility parameter, capturing the uncertainty on asset returns.

Solving the stochastic differential equation

$$R_t = log\frac{P_t}{P_{t-1}} = \left(\mu - \frac{\sigma^2}{2}t\right) + \sigma W_t \tag{3.23}$$

is the model obtained for log-returns. Therefore, it is possible to generate the scenario tree assuming that asset prices follow this stochastic process, using Monte Carlo simulation approach, starting from P_0 and proceeding through

$$P_{i,t} = P_{i,t-1} \exp\left((\mu - \frac{\sigma^2}{2}t) + \sigma W_t\right), \tag{3.24}$$

where:

- $P_{n,t}$ is the node n of the tree at time t;
- $P_{p(n),t-1}$ is the parent of the node n;
- W_t is sampled from a standard Brownian Motion.

Given the tree branching factors, through sampling it is possible to build the scenario tree needed for the successive optimization assigning uniform probability to each generated node.

3.5.3 Implementation

Below (Algorithm 1) is the pseudo-code for generating a defined number of children starting from a given parent node under the assumption that asset returns follow a Geometric Brownian Motion. The parameters $\boldsymbol{\mu}$, $\boldsymbol{\sigma}$ and ρ are estimated from historical data and represent mean, standard deviation and correlation matrix of asset log-returns, respectively. It is worth emphasizing that, to create the entire scenario tree, this function is iteratively called from the `ScenarioTree` class for each node that has children, so continuing until all leaves are generated.

Algorithm 1 Monte Carlo simulation for generating a period of the tree following the Geometric Brownian Motion.

procedure SIMULATE_ONE_TIME_STEP(S, `parent_node`)
    ▷ S is the number of children nodes to generate
    ▷ `p(n)` is the price vector corresponding to the parent node
    `arb` ← True
    ▷ `arb` is a boolean variable: 0 if there is not arbitrage, 1 otherwise
    `counter` ← 0
    while `arb` **and** `counter` < 100 **do**
        `counter` ← `counter` $+ 1$
        $\mathtt{B} \sim \mathcal{N}(\mathbf{0}, \rho)$ ▷ $\mathtt{B} \in \mathbb{R}^{M \times S}$
        $\mathtt{Y} \leftarrow \sigma \cdot \sqrt{dt} \cdot \mathtt{B}$
        $\mathtt{c} \leftarrow \mu - 0.5 \cdot \sigma^2$
        `returns` ← $\mathtt{c} \cdot \mathrm{dt} + \mathtt{Y}$
        for $i \leftarrow 0$ **to** M **do**
            for $s \leftarrow 0$ **to** S **do**
                `prices`$[i, s]$ ← `parent_node`$[i] \cdot \exp($`returns`$[i, s])$
            end for
        end for
        `arb` ← `check_arbitrage`(`prices`, `p(n)`)
    end while
    if `counter` ≥ 100 **then**
        raise Error("No arbitrage solution NOT found after 100 iterations")
    else
        `probs` ← $\frac{1}{S} \times \mathbf{1}_S$
        return `probs`, `prices`
    end if
end procedure

Chapter 4

L-Shaped decomposition

In [10] Benders presented a decomposition method to tackle mixed-integer linear programming when the technological matrix of the model has a suitable form that allows to separate the real variables from the integer variables. It consists in splitting the problem into a pure integer programming problem and a pure linear programming problem. These two problems are easier to solve then the original one and have to be solved iteratively in turn to arrive at the overall problem solution [11]. Due to the greater simplicity of the two new problems compared to the original mixed-integer problem, this approach is often more efficient than solving without decomposition, even though it requires multiple iterations. Stochastic programming problems inherently lend themselves to being approached by decomposition, since future decisions are easier to take once previous decisions are fixed and the realization of uncertainty is disclosed. Therefore, Benders decomposition has also been used to solve stochastic programming problems, under the name of L-shaped decomposition, firstly introduced in [12]. This Chapter depicts all the theoretical aspects needed to understand how the L-shaped approach works for the stochastic portfolio optimization problem, even in a multistage setting. Firstly, Section 4.1 contains a general discussion about L-shaped decomposition for two stage linear programming problems. Then the general model for portfolio optimization considered in this thesis is presented in Section 4.2. In Section 4.3 it is shown how the approach works for a two-stage portfolio optimization, considering different linear utility functions. Finally, in Section 4.4 the general setting of nested L-shaped decomposition to tackle multistage problems is explained.

4.1 L-shaped decomposition approach for general two-stage LP

A stochastic two-stage Linear Programming (LP) with recourse is a linear optimization problem that involve here-and-now decisions and second-stage actions that depend on the realization of uncertainty and on the decisions taken in the first-stage:

$$\begin{aligned} \min \quad & \mathbf{c}^\top \boldsymbol{x} + \mathbb{E}_\xi \left[Q(\boldsymbol{x}, \xi) \right] && (4.1) \\ \text{s.t.} \quad & A\boldsymbol{x} = \mathbf{b} && (4.2) \\ & \mathbf{x} \geq 0 && (4.3) \end{aligned}$$

is the general structure, also called deterministic equivalent of the stochastic problem, since the expected value of the objective function is considered. The second-stage problem is

$$Q\left(\boldsymbol{x}, \xi\right) \equiv \min_{\mathbf{y}} \{ \mathbf{d}^\top \mathbf{y} | W(\xi)\mathbf{y} = \mathbf{h}(\xi) - T(\xi)\boldsymbol{x}, \mathbf{y} \geq 0 \}. \tag{4.4}$$

Hence, $\mathbf{x}$ is the first-stage decision variable and $\mathbf{y}$ is the second-stage decision variable, ξ is the realization of uncertainty. If W depends on ξ, then the problem is classified as random recourse, otherwise it is called fixed recourse. $\mathcal{Q}(\boldsymbol{x}) = \mathbb{E}_\xi \left[Q(\boldsymbol{x}, \xi) \right]$ is called recourse function. By LP (strong) duality, it holds that

$$Q\left(\boldsymbol{x}, \xi\right) \equiv \max_{\boldsymbol{\lambda}} \{ \left(\mathbf{h}(\xi) - T(\xi)\boldsymbol{x} \right)^\top \lambda | W(\xi)^\top \lambda \leq \mathbf{d} \}. \tag{4.5}$$

It could be proved that the recourse function $\mathcal{Q}$ is convex. Now, let $\mathcal{S}$ be the set of scenarios that represent uncertainty in the second stage (and denote with $\boldsymbol{\pi}$ their probability) and consider

$$\begin{aligned} \min \quad & \mathbf{c}^\top \boldsymbol{x} + \sum_{s \in \mathcal{S}} \pi^{[s]} \mathbf{d}^{[s]\top} \mathbf{y}^{[s]} && && (4.6) \\ \text{s.t.} \quad & A\boldsymbol{x} = \mathbf{b} && && (4.7) \\ & W^{[s]}\mathbf{y}^{[s]} = \mathbf{h}^{[s]} - T^{[s]}\boldsymbol{x} && \forall s \in S && (4.8) \\ & \boldsymbol{x}, \mathbf{y}^{[s]} \geq 0, && && (4.9) \end{aligned}$$

that is the discretization of the problem (4.1) - (4.3). It is possible to note that for a given first-stage decision, second-stage problems are independent. Hence, the LP problem could be rewritten, through an usual trick, as

$$\begin{aligned} \min \quad & \mathbf{c}^\top \boldsymbol{x} + \Theta && (4.10) \\ \text{s.t.} \quad & A\boldsymbol{x} = \mathbf{b} && (4.11) \\ & \Theta \geq \mathcal{Q}(\boldsymbol{x}) && (4.12) \\ & \boldsymbol{x} \geq 0. && (4.13) \end{aligned}$$

The idea is to solve this problem, which is called master problem and involves only first-stage decision variable, relaxing it by approximating the constraint (4.12) using cutting planes, whose coefficients are obtained solving scenario subproblems for a given first-stage decision. Let $\hat{\boldsymbol{x}}$ be an optimal solution of the master problem and consider the dual of the second-stage problem for a given scenario $s \in \mathcal{S}$, which is

$$Q^{[s]}(\hat{\boldsymbol{x}}) \equiv \max_{\boldsymbol{\lambda}} \left(\mathbf{h}^{[s]} - T^{[s]}\hat{\boldsymbol{x}}\right)^{\top} \boldsymbol{\lambda}^{[s]} \tag{4.14}$$

$$\text{s.t. } W^{[s]\top}\boldsymbol{\lambda}^{[s]} \leq \mathbf{d}^{[s]\top}. \tag{4.15}$$

Since $\hat{\boldsymbol{\lambda}}^{[s]}$ is the optimal dual solution corresponding to $\hat{\boldsymbol{x}}$, but not to a generic $\boldsymbol{x}$, given an optimal dual solution $\hat{\boldsymbol{\lambda}}^{[s]}$, it holds that

$$Q^{[s]}(\boldsymbol{x}) \geq \left(\mathbf{h}^{[s]} - T^{[s]}\boldsymbol{x}\right)^{\top} \hat{\boldsymbol{\lambda}}^{[s]}. \tag{4.16}$$

Summing the inequality (4.16) over all the scenarios

$$\mathcal{Q}(\boldsymbol{x}) = \sum_{s\in\mathcal{S}} \pi^{[s]} Q^{[s]}(\boldsymbol{x}) \geq \sum_{s\in\mathcal{S}} \pi^{[s]} \left(\mathbf{h}^{[s]} - T^{[s]}\boldsymbol{x}\right)^{\top} \hat{\boldsymbol{\lambda}}^{[s]} \tag{4.17}$$

is obtained. Hence, the following optimality cut is added to the relaxed master problem:

$$\Theta \geq \sum_{s\in\mathcal{S}} \pi^{[s]} \left(\mathbf{h}^{[s]} - T^{[s]}\boldsymbol{x}\right)^{\top} \hat{\boldsymbol{\lambda}}^{[s]} \tag{4.18}$$

4.2 The portfolio optimization model

To see how the L-Shaped approach works in a portfolio optimization setting, it is needed to disclose in detail the model considered. Taking inspiration from [13], in this thesis a dynamic portfolio optimization model over a finite horizon $[0, T]$ is considered. As illustrated in Chapter 3, the uncertainty regarding future asset prices is approximated by a discrete distribution represented by a scenario tree, where each node is a realization of asset prices vector. The model includes purchase and sale variables for each risky asset and a riskless asset as liquidity component of the model on which risk-free interest is paid. The investor can rebalance the portfolio at finite dates: At the beginning of each period the decision maker observes the current market prices for the risky assets and revises the portfolio composition considering also conditional probabilities of future nodes. Transaction costs are considered both in selling and buying. The model includes also restrictions on short selling and borrowing. It is useful to introduce some notation regarding sets, functions, parameters and decision variables used in the model. Sets notation recalls what has been already introduced:

- $\mathcal{N}$: the set of nodes in the scenario tree;
- $\mathcal{N}_T$: the set of leaves (last period nodes) in the scenario tree;
- $I = \{1, ..., I\}$: the set of available assets.

Functions used are:

- $p(n)$: the parent node of node $n \in \mathcal{N}$;
- $u(\cdot)$: a concave utility function that reflects investor's risk aversion.

Decisions variables of the model are:

- $\boldsymbol{x}^{[n]} \in \mathbb{R}^I$: the portfolio composition at node n;
- $w^{[n]} \in \mathbb{R}$: the liquidity owned at node n (bank account);
- $\boldsymbol{b}^{[n]} \in \mathbb{R}^I$: the vector whose entries indicate the quantity of each asset $i \in I$ bought at node n;
- $\boldsymbol{s}^{[n]} \in \mathbb{R}^I$: the vector whose entries indicate the quantity of each asset $i \in I$ sold at node n;
- $R^{[n]}$: the investor total wealth at node n.

Finally, the parameters of the model are:

- $\bar{\boldsymbol{x}} \in \mathbb{R}^I$: the initial portfolio composition (before first actions are taken);
- $\bar{w} \in \mathbb{R}$: the initial liquidity position;
- $\pi^{[n]} \in \mathbb{R}^{\|\mathcal{N}\|}$: the probability associated with node n;
- $\boldsymbol{p}^{[n]} \in \mathbb{R}^I$: the asset prices' vector at node n;
- r_f: the risk-free return rate;
- c_b: the transaction cost for buying;
- c_s: the transaction cost for selling.

The multistage optimization model considered in this thesis for portfolio optimization is:

$$\max_{\boldsymbol{x},\boldsymbol{b},\boldsymbol{s},w} \quad \sum_{n\in\mathcal{N}_T} \pi^{[n]} u\left(R^{[n]}\right) \tag{4.19}$$

$$\text{s.t.} \quad \boldsymbol{x}^{[0]} = \bar{\boldsymbol{x}} + \boldsymbol{b}^{[0]} - \boldsymbol{s}^{[0]} \tag{4.20}$$

$$\boldsymbol{x}^{[n]} = \boldsymbol{x}^{[p(n)]} + \boldsymbol{b}^{[n]} - \boldsymbol{s}^{[n]}, \qquad \forall n \in \mathcal{N} \setminus \{0\} \tag{4.21}$$

$$w^{[0]} = \bar{w} + (1-c_s)\boldsymbol{p}^{[0]\top}\boldsymbol{s}^{[0]} - (1+c_b)\boldsymbol{p}^{[0]\top}\boldsymbol{b}^{[0]} \tag{4.22}$$

$$w^{[n]} = (1+r_f)w^{[p(n)]} + (1-c_s)\boldsymbol{p}^{[n]\top}\boldsymbol{s}^{[n]} - (1+c_b)\boldsymbol{p}^{[n]\top}\boldsymbol{b}^{[n]} \qquad \forall n \in \mathcal{N} \setminus \{0\} \tag{4.23}$$

$$R^{[n]} = w^{[n]} + \boldsymbol{p}^{[n]\top}\boldsymbol{x}^{[n]} \qquad \forall n \in \mathcal{N} \tag{4.24}$$

$$\boldsymbol{x}^{[n]}, \boldsymbol{b}^{[n]}, \boldsymbol{s}^{[n]}, w^{[n]} \geq 0 \qquad \forall n \in \mathcal{N} \tag{4.25}$$

The objective function (4.19) is the expected utility given the investor wealth at the end of the time horizon. Constraints (4.20) and (4.21) express the portfolio evolution (at node 0 and future nodes, respectively) according to the stocks purchased and sold. Constraints (4.22) and (4.23) state the bank account evolution considering the risk-free rate paid on the liquidity from the previous period and the cash flow due to shares trade (considering transaction costs). The wealth of the investor is given by its liquidity plus its portfolio value, as stated by Equation (4.24). Short-selling assets or borrowing money is not allowed, which is stated by constraint (4.25). Problem's characteristics made the model suitable to be solved through nested L-shaped decomposition, dividing the complete multistage model into smaller models that look at a single time window, but receive information from the other sub-models about the possible future effects of the chosen actions.

4.3 L-shaped decomposition approach for two-stage portfolio optimization

Since nested L-shaped, as suggested by its name, is simply an iterative application of the two-stage L-shaped decomposition, it is worth focusing on how the L-shaped approach works considering the case of two stages ($t = 0$, $t = 1$). Consider a simple situation as the one depicted in Figure 4.1. Let denote with $\boldsymbol{x}^{[s]}, \boldsymbol{b}^{[s]}, \boldsymbol{s}^{[s]}, w^{[s]}$ the second-stage decision variables, for $s \in \mathcal{S}$, where $\mathcal{S}$ is the set of scenarios ($\mathcal{S} = \{A, B\}$ in the Figure 4.1). The general problem introduced above may be

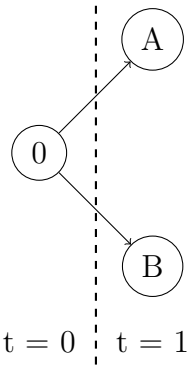

Figure 4.1: Example of a two-stages scenario tree

specifically rewritten for the two-stage case, as

$$\max_{\boldsymbol{x},\boldsymbol{b},\boldsymbol{s},w} \quad \sum_{s\in\mathcal{S}} \pi^{[s]} u\left(R^{[s]}\right) \tag{4.26}$$

$$\text{s.t.} \quad \boldsymbol{x}^{[0]} = \bar{\boldsymbol{x}} + \boldsymbol{b}^{[0]} - \boldsymbol{s}^{[0]} \tag{4.27}$$

$$w^{[0]} = \bar{w} + (1-c_s)\boldsymbol{p}^{[0]\top}\boldsymbol{s}^{[0]} - (1+c_b)\boldsymbol{p}^{[0]\top}\boldsymbol{b}^{[0]} \tag{4.28}$$

$$\boldsymbol{x}^{[s]} = \boldsymbol{x}^{[0]} + \boldsymbol{b}^{[s]} - \boldsymbol{s}^{[s]} \qquad \forall s \in \mathcal{S} \tag{4.29}$$

$$w^{[s]} = (1+r_f)w^{[0]} + (1-c_s)\boldsymbol{p}^{[s]\top}\boldsymbol{s}^{[s]} - (1+c_b)\boldsymbol{p}^{[s]\top}\boldsymbol{b}^{[s]} \qquad \forall s \in \mathcal{S} \tag{4.30}$$

$$R^{[s]} = w^{[s]} + \boldsymbol{p}^{[s]\top}\boldsymbol{x}^{[s]} \qquad \forall s \in \mathcal{S} \tag{4.31}$$

$$\boldsymbol{x}^{[0]}, \boldsymbol{b}^{[0]}, \boldsymbol{s}^{[0]}, w^{[0]} \geq 0 \tag{4.32}$$

$$\boldsymbol{x}^{[s]}, \boldsymbol{b}^{[s]}, \boldsymbol{s}^{[s]}, w^{[s]} \geq 0 \qquad \forall s \in \mathcal{S}. \tag{4.33}$$

To properly decompose the problem, it is important to note that constraints (4.27) and (4.28) involve only first-stage decision variables, while constraints (4.29) and (4.30) state the connection between first and second stage decision variables. Then, constraint (4.31) is the expression of the investor's wealth at the end of the time horizon given the portfolio composition and the liquidity position. Also, the objective function (4.26) depends only on the final wealth. Tackling the problem through L-shaped decomposition basically means to separate the problem into a master problem which involves the first-stage decision variables and a subproblem which involves the second-stage decision variables. The subproblem should be solved for each node belonging to the second stage (in Figure 4.1, nodes A and B), taking as parameters the first-stage solution found by solving the master problem.

4.3.1 Identity utility function

Let begin by considering the identity function as linear utility function, namely

$$u(R) = R \tag{4.34}$$

Using this utility function means that the investor has the unique aim of maximizing the expected total wealth at the end of the time horizon, unconcerned about risk. In fact, this utility function is a straight line, which is actually not concave (and neither convex), hence no risk aversion is expressed (as explained in Section 2.3). Rearranging the terms of the model (4.26)-(4.33) it is possible to separate the first-stage decision variables from the second-stage decision variables

$$\max_{\boldsymbol{x},\boldsymbol{b},\boldsymbol{s},w} \quad \sum_{s\in\mathcal{S}} \pi^{[s]} \left(w^{[s]} + {\boldsymbol{p}^{[s]}}^{\top} \boldsymbol{x}^{[s]} \right) \tag{4.35}$$

$$\text{s.t.} \quad \boldsymbol{x}^{[0]} - \boldsymbol{b}^{[0]} + \boldsymbol{s}^{[0]} = \bar{\boldsymbol{x}} \tag{4.36}$$

$$w^{[0]} - (1-c_s){\boldsymbol{p}^{[0]}}^{\top} \boldsymbol{s}^{[0]} + (1+c_b){\boldsymbol{p}^{[0]}}^{\top} \boldsymbol{b}^{[0]} = \bar{w} \tag{4.37}$$

$$\boldsymbol{x}^{[s]} - \boldsymbol{b}^{[s]} + \boldsymbol{s}^{[s]} = \boldsymbol{x}^{[0]} \qquad \forall s \in \mathcal{S} \tag{4.38}$$

$$w^{[s]} - (1-c_s){\boldsymbol{p}^{[s]}}^{\top} \boldsymbol{s}^{[s]} + (1+c_b){\boldsymbol{p}^{[s]}}^{\top} \boldsymbol{b}^{[s]} = (1+r_f) w^{[0]} \qquad \forall s \in \mathcal{S} \tag{4.39}$$

$$\boldsymbol{x}^{[0]}, \boldsymbol{b}^{[0]}, \boldsymbol{s}^{[0]}, w^{[0]} \geq 0 \tag{4.40}$$

$$\boldsymbol{x}^{[s]}, \boldsymbol{b}^{[s]}, \boldsymbol{s}^{[s]}, w^{[s]} \geq 0 \qquad \forall s \in \mathcal{S}. \tag{4.41}$$

Where the the definition of the wealth $R^{[s]}$ in Equation (4.31) is tacit since it can be embedded in the problem formulation as it has been done. With a slight abuse of notation the decision variables may be rewritten as vectors:

$$\begin{bmatrix} \boldsymbol{x}^{[0]} \\ \boldsymbol{b}^{[0]} \\ \boldsymbol{s}^{[0]} \\ w^{[0]} \end{bmatrix} \tag{4.42}$$

is the vector of first-stage decision variable,

$$\begin{bmatrix} \boldsymbol{x}^{[s]} \\ \boldsymbol{b}^{[s]} \\ \boldsymbol{s}^{[s]} \\ w^{[s]} \end{bmatrix} \tag{4.43}$$

is the vector of the second-stage decision variables. Using this notation and the identity function as utility function, it is possible to directly rewrite the model as a

discrete deterministic equivalent of a general two-stage LP (as in model (4.6)-(4.9))

$$\max_{\boldsymbol{x},\boldsymbol{b},\boldsymbol{s},w} \quad \sum_{s\in\mathcal{S}} \pi^{[s]} \begin{bmatrix} \boldsymbol{p}^{[s]} & 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} \boldsymbol{x}^{[s]} \\ \boldsymbol{b}^{[s]} \\ \boldsymbol{s}^{[s]} \\ w^{[s]} \end{bmatrix} \tag{4.44}$$

$$\text{s.t.} \quad \begin{bmatrix} 1 & -1 & 1 & 0 \\ 0 & (1+c_b)\boldsymbol{p}^{[0]\top} & -(1-c_s)\boldsymbol{p}^{[0]\top} & 1 \end{bmatrix} \begin{bmatrix} \boldsymbol{x}^{[0]} \\ \boldsymbol{b}^{[0]} \\ \boldsymbol{s}^{[0]} \\ w^{[0]} \end{bmatrix} = \begin{bmatrix} \bar{\boldsymbol{x}} \\ \bar{w} \end{bmatrix} \tag{4.45}$$

$$\begin{bmatrix} 1 & -1 & 1 & 0 \\ 0 & \frac{(1+c_b)}{(1+r_f)}\boldsymbol{p}^{[s]\top} & -\frac{(1-c_s)}{(1+r_f)}\boldsymbol{p}^{[s]\top} & \frac{1}{(1+r_f)} \end{bmatrix} \begin{bmatrix} \boldsymbol{x}^{[s]} \\ \boldsymbol{b}^{[s]} \\ \boldsymbol{s}^{[s]} \\ w^{[s]} \end{bmatrix} = \begin{bmatrix} \boldsymbol{x}^{[0]} \\ w^{[0]} \end{bmatrix} \tag{4.46}$$

$$\boldsymbol{x}^{[0]}, \boldsymbol{b}^{[0]}, \boldsymbol{s}^{[0]}, w^{[0]} \geq 0 \tag{4.47}$$

$$\boldsymbol{x}^{[s]}, \boldsymbol{b}^{[s]}, \boldsymbol{s}^{[s]}, w^{[s]} \geq 0 \qquad \forall s \in \mathcal{S}. \tag{4.48}$$

Following the notation used in model (4.6)-(4.9) it is possible to denote with

- $\boldsymbol{c} = \boldsymbol{0}$;
- $\boldsymbol{d}^\top = \begin{bmatrix} \boldsymbol{p}^{[s]} & 0 & 0 & 1 \end{bmatrix}$
- $A = \begin{bmatrix} 1 & -1 & 1 & 0 \\ 0 & (1+c_b)\boldsymbol{p}^{[0]\top} & -(1-c_s)\boldsymbol{p}^{[0]\top} & 1 \end{bmatrix}$;
- $b = \begin{bmatrix} \bar{\boldsymbol{x}} \\ \bar{w} \end{bmatrix}$;
- $W^{[s]} = \begin{bmatrix} 1 & -1 & 1 & 0 \\ 0 & \frac{(1+c_b)}{(1+r_f)}\boldsymbol{p}^{[s]\top} & -\frac{(1-c_s)}{(1+r_f)}\boldsymbol{p}^{[s]\top} & \frac{1}{(1+r_f)} \end{bmatrix}$;
- $h^{[s]} = \boldsymbol{0}$;
- $T^{[s]} = \begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix}$.

Hence, the technological matrix of the problem (4.44) - (4.48) has the form

$$\begin{bmatrix} A & 0 & 0 & \cdots & 0 \\ T^{[1]} & W^{[1]} & 0 & \cdots & 0 \\ T^{[2]} & 0 & W^{[2]} & \cdots & 0 \\ \vdots & \vdots & \vdots & \ddots & \vdots \\ T^{[S]} & 0 & 0 & \cdots & W^{[S]} \end{bmatrix}.$$

This form is called *dual block angular structure* and when the technological matrix of a two-stage problem has this form, it means that for given first-stage decisions, second-stage problems are independent. Hence, L-shaped decomposition can be applied. Following the same steps done in Section 4.1 it is possible to introduce an auxiliary variable Θ and rewrite the problem with respect to the first-stage decision variables only, as

$$\max_{\boldsymbol{x}^{[0]},\boldsymbol{b}^{[0]},\boldsymbol{s}^{[0]},w^{[0]}} \quad \Theta \tag{4.49}$$

$$\text{s.t.} \quad \begin{bmatrix} 1 & -1 & 1 & 0 \\ 0 & (1+c_b)\boldsymbol{p}^{[0]\top} & -(1-c_s)\boldsymbol{p}^{[0]\top} & 1 \end{bmatrix} \begin{bmatrix} \boldsymbol{x}^{[0]} \\ \boldsymbol{b}^{[0]} \\ \boldsymbol{s}^{[0]} \\ w^{[0]} \end{bmatrix} = \begin{bmatrix} \bar{\boldsymbol{x}} \\ \bar{w} \end{bmatrix} \tag{4.50}$$

$$\Theta \leq \mathcal{Q}\left(\boldsymbol{x}^{[0]}, w^{[0]}\right) \tag{4.51}$$

$$\boldsymbol{x}^{[0]}, \boldsymbol{b}^{[0]}, \boldsymbol{s}^{[0]}, w^{[0]} \geq 0 \tag{4.52}$$

Where $\mathcal{Q}\left(\boldsymbol{x}^{[0]}, w^{[0]}\right) = \mathbb{E}_{\boldsymbol{p}^{[s]}}\left[Q\left(x^{[0]}, w^{[0]}, \boldsymbol{p}^{[s]}\right)\right]$ is the recourse function, and

$$Q^{[s]}\left(\boldsymbol{x}^{[0]}, w^{[0]}, \boldsymbol{p}^{[s]}\right) \equiv \max_{\boldsymbol{x}^{[s]},\boldsymbol{b}^{[s]},\boldsymbol{s}^{[s]},w^{[s]}} \begin{bmatrix} \boldsymbol{p}^{[s]\top} & 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} \boldsymbol{x}^{[s]} \\ \boldsymbol{b}^{[s]} \\ \boldsymbol{s}^{[s]} \\ w^{[s]} \end{bmatrix} \tag{4.53}$$

$$\text{s.t.} \quad \begin{bmatrix} 1 & -1 & 1 & 0 \\ 0 & \frac{(1+c_b)}{(1+r_f)}\boldsymbol{p}^{[s]\top} & -\frac{(1-c_s)}{(1+r_f)}\boldsymbol{p}^{[s]\top} & \frac{1}{(1+r_f)} \end{bmatrix} \begin{bmatrix} \boldsymbol{x}^{[s]} \\ \boldsymbol{b}^{[s]} \\ \boldsymbol{s}^{[s]} \\ w^{[s]} \end{bmatrix} = \begin{bmatrix} \boldsymbol{x}^{[0]} \\ w^{[0]} \end{bmatrix} \tag{4.54}$$

$$\boldsymbol{x}^{[s]}, \boldsymbol{b}^{[s]}, \boldsymbol{s}^{[s]}, w^{[s]} \geq 0 \tag{4.55}$$

is the second-stage subproblem, given the first-stage post decision state variables $\left(\boldsymbol{x}^{[0]}, w^{[0]}\right)$ and the realization of uncertainty regarding asset prices $\left(\boldsymbol{p}^{[s]}\right)$.

Subproblem's dual and Benders' cuts

Given an optimal first-stage solution $\hat{\boldsymbol{x}}^{[0]}, \hat{w}^{[0]}$ and introducing the vector of dual variables $\begin{bmatrix} \boldsymbol{\lambda}^{[s]} \\ \sigma^{[s]} \end{bmatrix}$, exploiting LP strong duality, $Q\left(x^{[0]}, w^{[0]}, \boldsymbol{p}^{[s]}\right)$ can be restated

through its dual as

$$\min_{\boldsymbol{\lambda}^{[s]},\sigma^{[s]}} \quad \begin{bmatrix} \hat{\boldsymbol{x}}^{[0]\top} & \hat{w}^{[0]} \end{bmatrix} \begin{bmatrix} \boldsymbol{\lambda}^{[s]} \\ \sigma^{[s]} \end{bmatrix} \tag{4.56}$$

$$\text{s.t.} \quad \begin{bmatrix} 1 & 0 \\ -1 & \frac{(1+c_b)}{(1+r_f)}\boldsymbol{p}^{[s]\top} \\ 1 & -\frac{(1-c_s)}{(1+r_f)}\boldsymbol{p}^{[s]\top} \\ 0 & \frac{1}{(1+r_f)} \end{bmatrix} \begin{bmatrix} \boldsymbol{\lambda}^{[s]} \\ \sigma^{[s]} \end{bmatrix} \geq \begin{bmatrix} \boldsymbol{p}^{[s]\top} \\ 0 \\ 0 \\ 1 \end{bmatrix}. \tag{4.57}$$

Once found the optimal solution $\hat{\boldsymbol{\lambda}}^{[s]}, \hat{\sigma}^{[s]}$ for each scenario, the theoretical reasoning made in Section 4.1 leads to add

$$\Theta \leq \sum_{s\in\mathcal{S}} \pi^{[s]} \begin{bmatrix} \boldsymbol{x}^{[0]\top} & w^{[0]} \end{bmatrix} \begin{bmatrix} \hat{\boldsymbol{\lambda}}^{[s]} \\ \hat{\sigma}^{[s]} \end{bmatrix} \tag{4.58}$$

as optimality cuts to the relaxed master problem. It is worth saying that in general, when L-shaped approach is used, it could happen that for a given first-stage decision, a subproblem may be unfeasible and in such a case an infeasibility cut is added to the master problem [11]. However, this could not happen in the portfolio optimization model introduced, since there is no reason why a first-stage portfolio composition should lead to an infeasible subproblem.

Relaxed Master Problem

Hence, the relaxed master problem becomes

$$\max_{\boldsymbol{x}^{[0]},\boldsymbol{b}^{[0]},\boldsymbol{s}^{[0]},w^{[0]}} \quad \Theta \tag{4.59}$$

$$\text{s.t.} \quad \begin{bmatrix} 1 & -1 & 1 & 0 \\ 0 & (1+c_a)\boldsymbol{p}^{[0]\top} & -(1-c_v)\boldsymbol{p}^{[0]\top} & 1 \end{bmatrix} \begin{bmatrix} \boldsymbol{x}^{[0]} \\ \boldsymbol{b}^{[0]} \\ \boldsymbol{s}^{[0]} \\ w^{[0]} \end{bmatrix} = \begin{bmatrix} \bar{\boldsymbol{x}} \\ \bar{w} \end{bmatrix} \tag{4.60}$$

$$\Theta \leq \sum_{s\in\mathcal{S}} \pi^{[s]} \begin{bmatrix} \boldsymbol{x}^{[0]\top} & w^{[0]} \end{bmatrix} \begin{bmatrix} \hat{\boldsymbol{\lambda}}^{[s]} \\ \hat{\sigma}^{[s]} \end{bmatrix} \tag{4.61}$$

$$\boldsymbol{x}^{[0]}, \boldsymbol{b}^{[0]}, \boldsymbol{s}^{[0]}, w^{[0]} \geq 0. \tag{4.62}$$

Where the constraints of the form (4.61) are called Benders' cuts and they are iteratively added by solving the subproblems (for each scenario).

4.3.2 Piecewise linear utility function

Actually, there is a way to preserve the linearity of the problem without giving up the ability to express investors' usual risk aversion and that is to use a concave

piecewise linear function as utility function. To express risk aversion, losses should be penalized more than gains are rewarded, using two coefficients $l, v \geq 0 | l \geq v$. Let denote with $\bar{R}$ the initial wealth owned by the investor and consider

$$u\left(R^{[s]}\right) = \begin{cases} v\left(R^{[s]} - \bar{R}\right) & \text{if } R^{[s]} > \bar{R}, \\ l\left(R^{[s]} - \bar{R}\right) & \text{if } R^{[s]} < \bar{R}. \end{cases} \tag{4.63}$$

To properly define the linear optimization problem, it is needed to define two decision variables

$$r_+^{[s]} = \max\{0, \left(R^{[s]} - \bar{R}\right)\} \tag{4.64}$$

$$r_-^{[s]} = \max\{0, \left(\bar{R} - R^{[s]}\right)\}, \tag{4.65}$$

that represent the wealth surplus or shortfall in scenario s with respect to the initial wealth $\bar{R}$. In particular, $r_+^{[s]}$ represents the surplus (gains), while $r_-^{[s]}$ represents losses in scenario s with respect to the initial wealth owned. In this way, the utility function can be rewritten as

$$u\left(r_+^{[s]}, r_-^{[s]}\right) = v r_+^{[s]} - l r_-^{[s]}, \tag{4.66}$$

in order to keep the model linear. Hence the model can be stated in the classic form of a two-stage LP:

$$\max_{\boldsymbol{x},\boldsymbol{b},\boldsymbol{s},w} \quad \sum_{s\in\mathcal{S}} \pi_s \left(v r_+^{[s]} - l r_-^{[s]}\right) \tag{4.67}$$

$$\text{s.t.} \quad \boldsymbol{x}^{[0]} = \bar{\boldsymbol{x}} + \boldsymbol{b}^{[0]} - \boldsymbol{s}^{[0]} \tag{4.68}$$

$$w^{[0]} = \bar{w} + (1 - c_s)\boldsymbol{p}^{[0]\top}\boldsymbol{s}^{[0]} - (1 + c_b)\boldsymbol{p}^{[0]\top}\boldsymbol{b}^{[0]} \tag{4.69}$$

$$\boldsymbol{x}^{[s]} = \boldsymbol{x}^{[0]} + \boldsymbol{b}^{[s]} - \boldsymbol{s}^{[s]} \qquad \forall s \in \mathcal{S} \tag{4.70}$$

$$w^{[s]} = (1 + r_f)w^{[0]} + (1 - c_s)\boldsymbol{p}^{[s]\top}\boldsymbol{s}^{[s]} - (1 + c_b)\boldsymbol{p}^{[s]\top}\boldsymbol{b}^{[s]} \qquad \forall s \in \mathcal{S} \tag{4.71}$$

$$w^{[s]} + \boldsymbol{p}^{[s]}\boldsymbol{x}^{[s]} = \bar{R} + r_+^{[s]} - r_-^{[s]} \qquad \forall s \in \mathcal{S} \tag{4.72}$$

$$\boldsymbol{x}^{[0]}, \boldsymbol{b}^{[0]}, \boldsymbol{s}^{[0]}, w^{[0]} \geq 0 \tag{4.73}$$

$$\boldsymbol{x}^{[s]}, \boldsymbol{b}^{[s]}, \boldsymbol{s}^{[s]}, w^{[s]} \geq 0 \qquad \forall s \in \mathcal{S} \tag{4.74}$$

$$r_+^{[s]}, r_-^{[s]} \geq 0 \qquad \forall s \in \mathcal{S}. \tag{4.75}$$

Where the objective function (4.67) takes into consideration the piecewise linear utility function introduced before, using gains and losses variables that are defined through constraints (4.72) and (4.75). With a slight abuse of notation, the vectorial

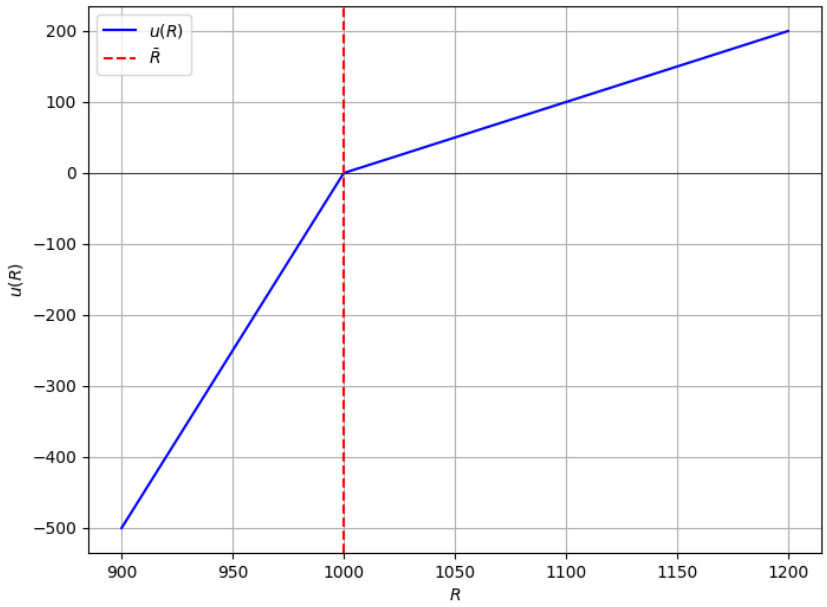

Figure 4.2: Example of piecewise linear utility function: $v = 1$, $l = 5$, $\bar{R} = 1000$

form of the problem is

$$\max_{\boldsymbol{x},\boldsymbol{b},\boldsymbol{s},w} \quad \sum_{s\in\mathcal{S}} \pi^{[s]} \begin{bmatrix} v & -l & 0 & 0 & 0 & 0 \end{bmatrix} \begin{bmatrix} r_+^{[s]} \\ r_-^{[s]} \\ \boldsymbol{x}^{[s]} \\ \boldsymbol{b}^{[s]} \\ \boldsymbol{s}^{[s]} \\ w^{[s]} \end{bmatrix} \tag{4.76}$$

$$\text{s.t.} \quad \begin{bmatrix} 0 & 0 & 1 & -1 & 1 & 0 \\ 0 & 0 & 0 & (1+c_b)\boldsymbol{p}^{[0]\top} & -(1-c_s)\boldsymbol{p}^{[0]\top} & 1 \end{bmatrix} \begin{bmatrix} r_+^{[s]} \\ r_-^{[s]} \\ \boldsymbol{x}^{[0]} \\ \boldsymbol{b}^{[0]} \\ \boldsymbol{s}^{[0]} \\ w^{[0]} \end{bmatrix} = \begin{bmatrix} \bar{\boldsymbol{x}} \\ \bar{w} \end{bmatrix} \tag{4.77}$$

$$\begin{bmatrix} 0 & 0 & 1 & -1 & 1 & 0 \\ 0 & 0 & 0 & \frac{(1+c_b)}{(1+r_f)}\boldsymbol{p}^{[s]\top} & -\frac{(1-c_s)}{(1+r_f)}\boldsymbol{p}^{[s]\top} & \frac{1}{(1+r_f)} \\ -1 & 1 & \boldsymbol{p}^{[s]\top} & 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} r_+^{[s]} \\ r_-^{[s]} \\ \boldsymbol{x}^{[s]} \\ \boldsymbol{b}^{[s]} \\ \boldsymbol{s}^{[s]} \\ w^{[s]} \end{bmatrix} = \begin{bmatrix} \boldsymbol{x}^{[0]} \\ w^{[0]} \\ \bar{R} \end{bmatrix} \tag{4.78}$$

$$\boldsymbol{x}^{[0]}, \boldsymbol{b}^{[0]}, \boldsymbol{s}^{[0]}, w^{[0]} \geq 0 \tag{4.79}$$

$$\boldsymbol{x}^{[s]}, \boldsymbol{b}^{[s]}, \boldsymbol{s}^{[s]}, w^{[s]} \geq 0 \qquad \forall s \in \mathcal{S} \tag{4.80}$$

$$r^+, r^- \geq 0. \tag{4.81}$$

The technological matrix has the dual block angular structure, so L-shaped decomposition is possible. Proceeding as above, one can introduce Θ, the recourse function $\mathcal{Q}\left(\boldsymbol{x}^{[0]}, w^{[0]}, \bar{R}\right) = \mathbb{E}_{\boldsymbol{p}^{[s]}}\left[Q\left(x^{[0]}, w^{[0]}, \bar{R}, \boldsymbol{p}^{[s]}\right)\right]$ and write the master problem

$$\max_{\boldsymbol{x}^{[0]}, \boldsymbol{b}^{[0]}, \boldsymbol{s}^{[0]}, w^{[0]}} \quad \Theta \tag{4.82}$$

$$\text{s.t.} \quad \begin{bmatrix} 1 & -1 & 1 & 0 \\ 0 & (1+c_b)\boldsymbol{p}^{[0]\top} & -(1-c_s)\boldsymbol{p}^{[0]\top} & 1 \end{bmatrix} \begin{bmatrix} \boldsymbol{x}^{[0]} \\ \boldsymbol{b}^{[0]} \\ \boldsymbol{s}^{[0]} \\ w^{[0]} \end{bmatrix} = \begin{bmatrix} \bar{\boldsymbol{x}} \\ \bar{w} \end{bmatrix} \tag{4.83}$$

$$\Theta \leq \mathcal{Q}\left(\boldsymbol{x}^{[0]}, w^{[0]}, \bar{R}\right) \tag{4.84}$$

$$\boldsymbol{x}^{[0]}, \boldsymbol{b}^{[0]}, \boldsymbol{s}^{[0]}, w^{[0]} \geq 0. \tag{4.85}$$

Where $Q^{[s]}\left(\boldsymbol{x}^{[0]}, w^{[0]}, \bar{R}, \boldsymbol{p}^{[s]}\right)$ is defined as

$$\max_{\boldsymbol{x}^{[s]}, \boldsymbol{b}^{[s]}, \boldsymbol{s}^{[s]}, w^{[s]}} \quad \begin{bmatrix} v & -l & 0 & 0 & 0 & 0 \end{bmatrix} \begin{bmatrix} r_+^{[s]} \\ r_-^{[s]} \\ \boldsymbol{x}^{[s]} \\ \boldsymbol{b}^{[s]} \\ \boldsymbol{s}^{[s]} \\ w^{[s]} \end{bmatrix} \tag{4.86}$$

$$\text{s.t.} \quad \begin{bmatrix} 0 & 0 & 1 & -1 & 1 & 0 \\ 0 & 0 & 0 & \frac{(1+c_b)}{(1+r_f)}\boldsymbol{p}^{[s]\top} & -\frac{(1-c_s)}{(1+r_f)}\boldsymbol{p}^{[s]\top} & \frac{1}{(1+r_f)} \\ -1 & 1 & \boldsymbol{p}^{[s]\top} & 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} r_+^{[s]} \\ r_-^{[s]} \\ \boldsymbol{x}^{[s]} \\ \boldsymbol{b}^{[s]} \\ \boldsymbol{s}^{[s]} \\ w^{[s]} \end{bmatrix} = \begin{bmatrix} \boldsymbol{x}^{[0]} \\ w^{[0]} \\ \bar{R} \end{bmatrix} \tag{4.87}$$

$$\boldsymbol{x}^{[s]}, \boldsymbol{b}^{[s]}, \boldsymbol{s}^{[s]}, w^{[s]} \geq 0 \tag{4.88}$$

$$r^+, r^- \geq 0. \tag{4.89}$$

Subproblem's dual and Benders' cuts

The problem is still linear, so strong duality still holds. In fact, introducing the vector of dual variables $\begin{bmatrix} \boldsymbol{\lambda}^{[s]} \\ \sigma^{[s]} \\ \gamma^{[s]} \end{bmatrix}$, given $\hat{\boldsymbol{x}}^{[0]}, \hat{w}^{[0]}$ and $\bar{R}$, the second-stage subproblem

may be rewritten through its dual as

$$\min_{\boldsymbol{\lambda}^{[s]},\sigma^{[s]},\gamma^{[s]}} \quad \begin{bmatrix}\hat{\boldsymbol{x}}^{[0]\top} & \hat{w}^{[0]} & \bar{R}\end{bmatrix}\begin{bmatrix}\boldsymbol{\lambda}^{[s]}\\ \sigma^{[s]}\\ \gamma^{[s]}\end{bmatrix} \tag{4.90}$$

$$\text{s.t.} \quad \begin{bmatrix} 0 & 0 & -1\\ 0 & 0 & 1\\ 1 & 0 & \boldsymbol{p}^{[s]\top}\\ -1 & \frac{(1+c_b)}{(1+r_f)}\boldsymbol{p}^{[s]\top} & 0\\ 1 & -\frac{(1-c_s)}{(1+r_f)}\boldsymbol{p}^{[s]\top} & 0\\ 0 & \frac{1}{(1+r_f)} & 1\end{bmatrix}\begin{bmatrix}\boldsymbol{\lambda}^{[s]}\\ \sigma^{[s]}\\ \gamma^{[s]}\end{bmatrix} \geq \begin{bmatrix} v\\ -l\\ 0\\ 0\\ 0\\ 0\end{bmatrix}. \tag{4.91}$$

Proceeding in the same way, the optimal dual variables $\hat{\boldsymbol{\lambda}}^{[s]}, \hat{\sigma}^{[s]}, \hat{\gamma}^{[s]}$ are used to add

$$\Theta \leq \sum_{s\in\mathcal{S}} \pi^{[s]} \begin{bmatrix}\boldsymbol{x}^{[0]\top} & w^{[0]} & \bar{R}\end{bmatrix}\begin{bmatrix}\hat{\boldsymbol{\lambda}}^{[s]}\\ \hat{\sigma}^{[s]}\\ \hat{\gamma}^{[s]}\end{bmatrix} \tag{4.92}$$

as optimality cuts to the master problem.

Relaxed master problem

Finally, the relaxed master problem have the following form

$$\max_{\boldsymbol{x}^{[0]},\boldsymbol{b}^{[0]},\boldsymbol{s}^{[0]},w^{[0]}} \quad \Theta \tag{4.93}$$

$$\text{s.t.} \quad \begin{bmatrix}1 & -1 & 1 & 0\\ 0 & (1+c_b)\boldsymbol{p}^{[0]\top} & -(1-c_s)\boldsymbol{p}^{[0]\top} & 1\end{bmatrix}\begin{bmatrix}\boldsymbol{x}^{[0]}\\ \boldsymbol{b}^{[0]}\\ \boldsymbol{s}^{[0]}\\ w^{[0]}\end{bmatrix} = \begin{bmatrix}\bar{\boldsymbol{x}}\\ \bar{w}\end{bmatrix} \tag{4.94}$$

$$\Theta \leq \sum_{s\in\mathcal{S}} \pi^{[s]} \begin{bmatrix}\boldsymbol{x}^{[0]\top} & w^{[0]} & \bar{R}\end{bmatrix}\begin{bmatrix}\hat{\boldsymbol{\lambda}}^{[s]}\\ \hat{\sigma}^{[s]}\\ \hat{\gamma}^{[s]}\end{bmatrix} \tag{4.95}$$

$$\boldsymbol{x}^{[0]}, \boldsymbol{b}^{[0]}, \boldsymbol{s}^{[0]}, w^{[0]} \geq 0. \tag{4.96}$$

4.4 Nested L-shaped

Nested L-shaped, as suggested by its name, consists in recursively applying the L-shaped decomposition across a multistage scenario tree, treating it as a series

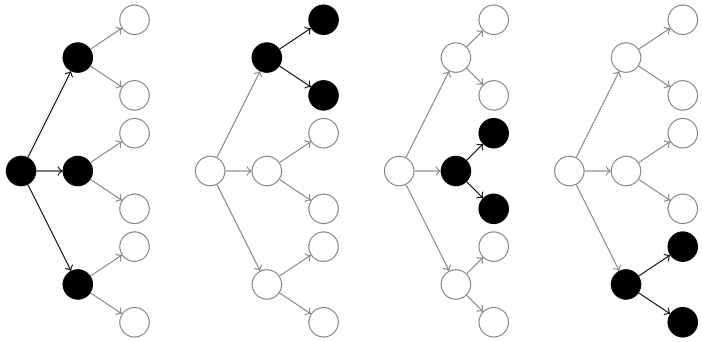

Figure 4.3: Illustration of the Nested L-Shaped approach

of nested two-stage problems. The core idea, illustrated in Figure 4.3, is to iteratively solve each two-stage sub-tree of the overall scenario tree using L-shaped decomposition. In this framework, each inner node (meaning all nodes except the root node and the leaves) plays a double role: as a child node, it generates multipliers to create Benders' cuts for its parent's problem, while as the root of its own two-stage sub-tree, it passes decisions downstream to its children. The process begins by solving the master problem at the root node, initially without any Benders' cuts. Subsequently, each second-stage node receives the solution from node 0 and solves its respective subproblem, passing the results to its own children. This process continues until the leaf nodes are reached. Once the leaves are reached, the process reverses, with each leaf solving its subproblem to derive the optimal dual variables, which are then passed back to its parent to introduce Benders' cuts. Moving backwards up the tree, each node solves its subproblem with the newly added cuts, determining the multipliers that will be passed to its parent for additional optimality cuts. When node 0 is reached again, a new iteration begins, with the master problem being solved using the updated Benders' cuts. The solution is then propagated forward through the tree, and the entire process repeats until convergence.

Chapter 5

Computational experiments

Several tests have been conducted to evaluate the end-to-end process, from scenario generation to solving the optimization model using L-shaped decomposition. The parameters required by the moment matching method and the Monte-Carlo sampling to generate the scenario tree were derived from real data using the Python library `yfinance`. Specifically, monthly log-returns were calculated using closing stock prices on the last day of each month, spanning from May 2023 to April 2024. Log-returns were chosen for their stability, which is advantageous compared to prices; after generating the log-returns for each scenario, the corresponding prices were computed. A monthly interval was selected as it offers a good balance between stability (with less noise than daily data) and a reasonable time horizon for the investment. For the L-shaped decomposition, master problems and subproblems were all solved with `Gurobi`.

5.1 First case: Two assets and two stages

The first experiment focuses on two assets and a two-stage scenario tree. Alphabet (GOOG) and Amazon (AMZN) were selected as the companies whose stocks are available in the market, along with the risk-free asset. The initial step involves gathering stock price data and extracting the necessary parameters to generate the scenario tree using the chosen methods. Table 5.1 presents the estimated moments for GOOG and AMZN assets' log-returns. The stock prices on the last day of April 2024 are used as prices for node 0 of the scenario tree. Hence, in this configuration, the second stage ($t = 1$) of the tree represents the potential market scenarios on the last day of May 2024.

Mean:	$\hat{\boldsymbol{\mu}} = \begin{bmatrix} 0.03386114 & 0.02623393 \end{bmatrix}$
Standard Deviation:	$\hat{\boldsymbol{\sigma}} = \begin{bmatrix} 0.05736424 & 0.05282871 \end{bmatrix}$
Skewness:	$\hat{\boldsymbol{t}} = \begin{bmatrix} -0.41745866 & -0.14091403 \end{bmatrix}$
Kurtosis:	$\hat{\boldsymbol{\kappa}} = \begin{bmatrix} 3.91793071 & 1.39701539 \end{bmatrix}$
Correlation:	$\hat{\rho} = \begin{bmatrix} 1 & -0.03443175 \\ -0.03443175 & 1 \end{bmatrix}$

Table 5.1: First case: Statistical moments estimated from real data for GOOG and AMZN assets' monthly log-returns.

5.1.1 Scenario generation

The MM algorithm uses all the estimated moments as inputs, aiming to replicate them when generating the scenario tree. For this instance, five scenarios were produced. Solving the MM optimization model (3.19)-(3.21) to generate the tree took a computational time of 0.211 seconds. Figure 5.1 illustrates the resulting tree. The solution of the moment matching model yielded to an optimal objective

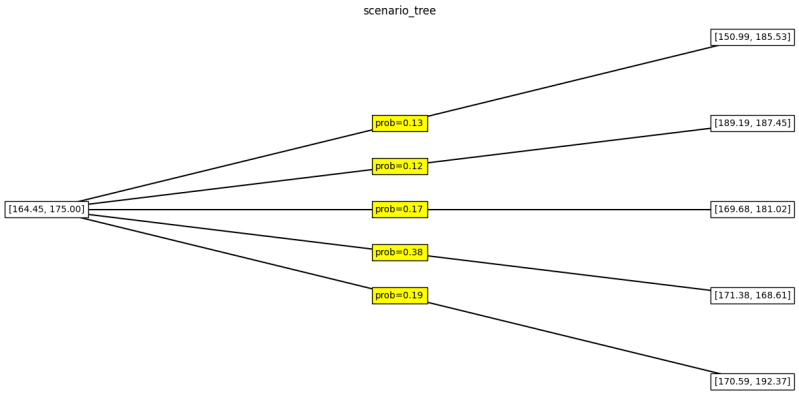

Figure 5.1: First case: Two-stage scenario tree of GOOG and AMZN stock prices, generated with the MM method

value of 2.487×10^{-5}, indicating that the moments of the generated tree deviate

only slightly from the expected moments, with a squared difference close to zero. For comparison, a tree was also generated using the Monte-Carlo sampling from GBM, which requires only the estimated mean, standard deviation, and correlation as input parameters for the simulation. The obtained tree is represented in Figure 5.2. In the GBM-generated tree, all scenarios are equally likely because they are

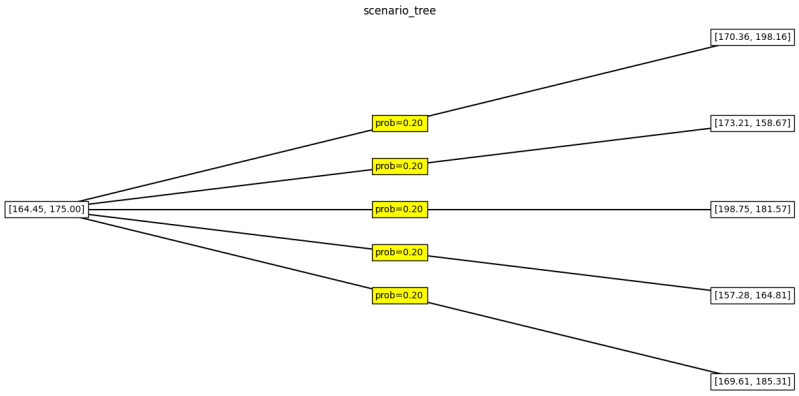

Figure 5.2: First case: Two-stage scenario tree of GOOG and AMZN stock prices, generated with GBM method

constructed through random sampling, making it impossible to assign different probabilities based on a defined logic. To assess the accuracy of both methods, the squared difference between the historical moments and those of the tree generated with the GBM method was calculated as well, yielding a value of 6.91, whereas it was approximately zero for the tree generated with the MM method. This result should not imply that moment matching is always the better choice. The choice between the two approaches should depend on several factors, particularly the available information about future asset behavior and the number of scenarios to generate. Generally, if an investor has some beliefs about future scenarios (such as expected moments) or other requirement that could be expressed as constraints in the MM optimization model, then the moment matching method would be the preferable choice. Conversely, if a large number of scenarios is required, Monte-Carlo simulation using the GBM method is likely more suitable, since sampling tends to perform better with more samples, while the computational complexity of the MM model increases significantly as the number of scenarios grows.

5.1.2 Model resolution

After the tree was built, the optimization model was solved through the L-Shaped decomposition, following the process explained in 4.3.2. The tree generated through the MM method (Figure 5.1) was the one considered. For the model defined in (4.67)-(4.75), the coefficients of the piecewise linear utility function were set to $v = 2$ (for gains) and $l = 5$ (for losses). The risk-free rate was fixed at $r_f = 0.1$, with proportional transaction costs of $c_s = 0.005$ for selling and $c_b = 0.005$ for buying. The initial liquidity was assumed to be 1000\$, the initial portfolio composition was assumed to be $\bar{\boldsymbol{x}} = \begin{bmatrix} 0 & 0 \end{bmatrix}$. The L-shaped algorithm converged to a solution after 4 iterations, requiring 0.098 seconds of computational time. The optimal portfolio $\boldsymbol{x}^{[0]^*}$ found is reported in Table 5.2. It resulted in an optimal objective

GOOG	**AMZN**	**liquidity**
3.19	2.68	0.00

Table 5.2: First case (MM) results: optimal solution

value of $U^* = 46.46$ (that is the expected utility corresponding to the optimal solution). Table 5.3 summarizes the results obtained emphasizing how the investor wealth changes based on the generated future scenarios at $t = 1$. To get a term of

Optimal portfolio composition	[3.19, 2.68]	
Node	**Prices (\$)**	**Investor Wealth (\$)**
Node 1	[150.99, 185.53]	980.32
Node 2	[189.19, 187.45]	1107.44
Node 3	[169.68, 181.02]	1027.89
Node 4	[171.38, 168.61]	1000.00
Node 5	[170.59, 192.37]	1061.27

Table 5.3: First case results: optimal portfolio composition and investor wealth for each scenario

comparison, the model was also solved by direct resolution (without decomposition) with `Gurobi`. After a computational time of 0.0058 seconds, the same solution was found, with the same optimal value for the objective function. It is worth emphasizing the fact that the portfolio is actually not modified in the leaf nodes, in fact investor wealth can not be improved in the last stage by modifying its position on the market, since the portfolio should be *self-financing*. On the contrary selling

and buying shares would have a negative impact on the investor wealth, due to the transaction costs.
To complete the comparison of the previous paragraph, the portfolio optimization model with the same parameters was also solved over the tree obtained with the GBM method (illustrated in Figure 5.2) and returns the solution shown in Table 5.4.

GOOG	AMZN	liquidity
6.05	0.00	0.00

Table 5.4: First case (GBM) results: optimal solution

5.2 Second case: Five assets and two stages

Once assessed that the process works, it is the case to enlarge the dimension of the market to see what happens. The following list reports the five companies considered in the market (along with the risk-free asset) and their corresponding sector:

- Exxon Mobil Corp. (XOM): Energy;
- Walmart Inc. (WMT): Retail;
- The Coca-Cola Company (KO): Beverages;
- Alphabet Inc. (GOOG): Internet Services;
- Amazon.com Inc. (AMZN): Technology and E-commerce;

5.2.1 Scenario generation

The tree was built with the moment matching method, choosing 10 as number of children. After 3 iterations the MM optimization model leaded to an optimal free-of-arbitrage tree, corresponding to an optimal value of 0.001 in the squared distance among the statistical moments of the tree and the estimated moments. An iteration here is intended as a complete resolution of the moment matching model, which may lead to a tree that allow the presence of arbitrage opportunities; in such a case the model is solved again (and the counter of iterations is increased by one), until an arbitrage free solution is found.

5.2.2 Model resolution

Using the same parameters of the previous section for the model, the optimal portfolio composition found by solving the model through the L-Shaped decomposition is shown in Table 5.5. The L-Shaped algorithm required 9 iterations to converge

XOM	WMT	KO	GOOG	AMZN	liquidity
3.86	8.34	0	0	0.25	0.00

Table 5.5: Second case results: optimal solution

and a computational time of 0.371 seconds. The expected utility given the optimal solution was $U^* = 38.89$. A direct resolution leads to the same results in 0.0386 seconds.
To better understand the role played by the utility function and the risk aversion, the model was solved again with the same setting, but increasing the coefficient l from 5 to 7, reflecting a major risk aversion. This leaded to a new portfolio

XOM	WMT	KO	GOOG	AMZN	liquidity
2.32	4.55	0	0	0.40	382.59

Table 5.6: Second case results: optimal solution with a major risk aversion

composition (shown in Table 5.6) where liquidity position is increased in spite of positions on shares, as the bank account provides a guaranteed income, so the portfolio risk is reduced. Reducing the portfolio risk means to give up some potential gains to avoid some potential losses, as one can see in Figures 5.3 and 5.4 which show the distribution of investor wealth over the ten scenarios of the tree.

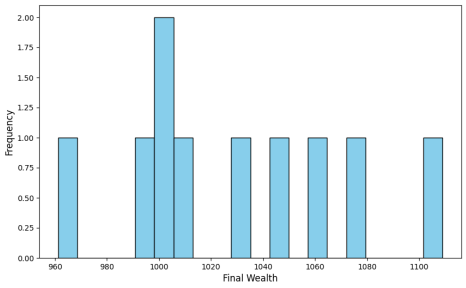

Figure 5.3: Second case: Investor wealth distribution over scenarios, for the model with $v = 2$ and $l = 5$ (less risk aversion)

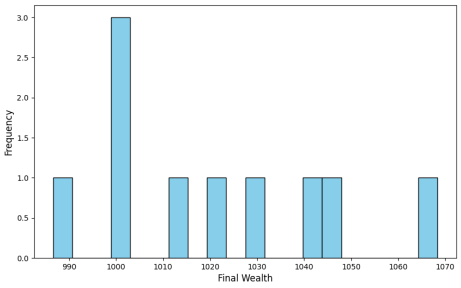

Figure 5.4: Second case: Investor wealth distribution over scenarios, for the model with $v = 2$ and $l = 7$ (more risk aversion)

Furthermore, if the model was solved with the identity function as utility function, as outlined in Section 4.3.1, then the optimal portfolio obtained is the one described in Table 5.7. The portfolio would be completely unbalanced towards an asset. This

XOM	WMT	KO	GOOG	AMZN	liquidity
8.41316374	0	0	0	0	0

Table 5.7: Second case results: optimal solution if the identity function was the utility function

is because the identity function does not express any risk aversion, leading to all liquidity being allocated to a long position in the asset with the highest expected return in future scenarios. In fact, *portfolio diversification*, which involves spreading investments across various assets to reduce risk exposure, is a common strategy for managing risk. However, without risk aversion, diversification loses its relevance since there is no incentive to mitigate risk. Looking at the model (4.35)-(4.41) it becomes clear that using the identity function in this setting means solely aiming to maximize the expected final wealth. Therefore, it is reasonable to concentrate all the liquidity on the single asset with the highest expected return.

5.3 Third case: Two assets and three stages

The third case focused on a three stages tree and a market composed by stocks of Alphabet (GOOG) and Amazon (AMZN) along with the risk-free asset. The

branching factors selected for this case were 5 for node 0 and 4 for the nodes belonging to the second stage.

5.3.1 Scenario generation

In order to generate a multistage scenario tree (as outlined in Section 3.4.2) multiple resolutions of the moment matching method are required, precisely one for each two-stage subtree. In particular, the first solution leads to build the second layer ($t = 1$) of the tree. Then, the moment matching method is performed again once for each node that belongs to the second stage to generate its own children that compose the third layer of the tree. Hence, in this case the moment matching method was performed 6 times to create the entire tree and for each of these times, multiple iterations could be required to get an arbitrage-free solution. This leads to a computational time of 0.93 seconds to build the tree. Estimated statistical moments were used as expected moments both for the generation of second and third stage. In real applications, if a domain expert has some information or beliefs regarding shares' behavior, then different expected moments can be used to generate different periods of the tree, even making them depend on the outcome of the previous stage. In this instance, the optimal tree found with the moment matching method leads to an average squared difference among the expected moments and the moments underlying the optimal tree of 0.04. The scenario tree obtained is shown in Figure 5.5. For this case the scenario tree was also generated through

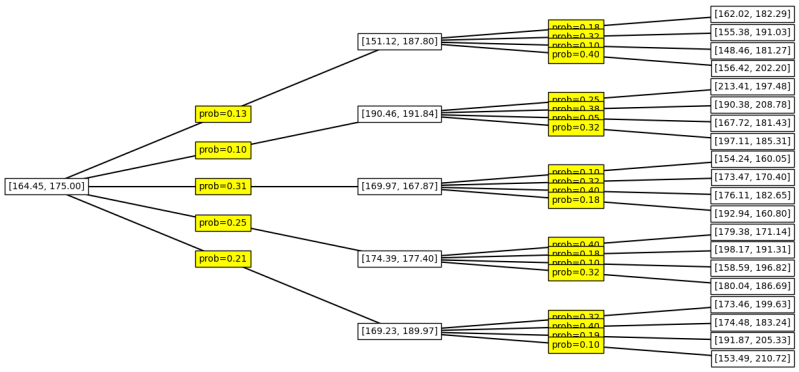

Figure 5.5: Third case: Three-stage scenario tree of GOOG and AMZN stocks prices, generated with the Moment Matching method

the GBM method, which leaded to the tree shown in Figure 5.6. As expected, tree generation with this method was faster (0.27 seconds).

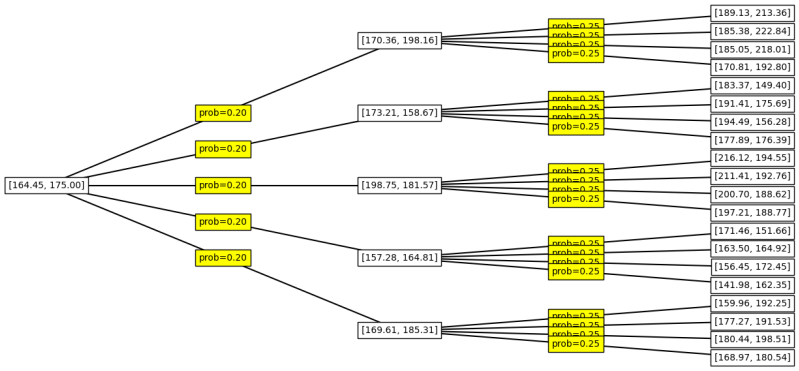

Figure 5.6: Third case: Three-stage scenario tree of GOOG and AMZN stocks prices, generated with the Geometric Brownian Motion method

Figure 5.7 illustrates the real evolution of assets prices together with the different scenarios generated with the moment matching method. Figure 5.8 does the same for scenarios generated through sampling from Geometric Brownian Motion.

Figure 5.7: Third case: MM scenarios versus real evolution of shares prices (dimension of nodes are proportional to their probability)

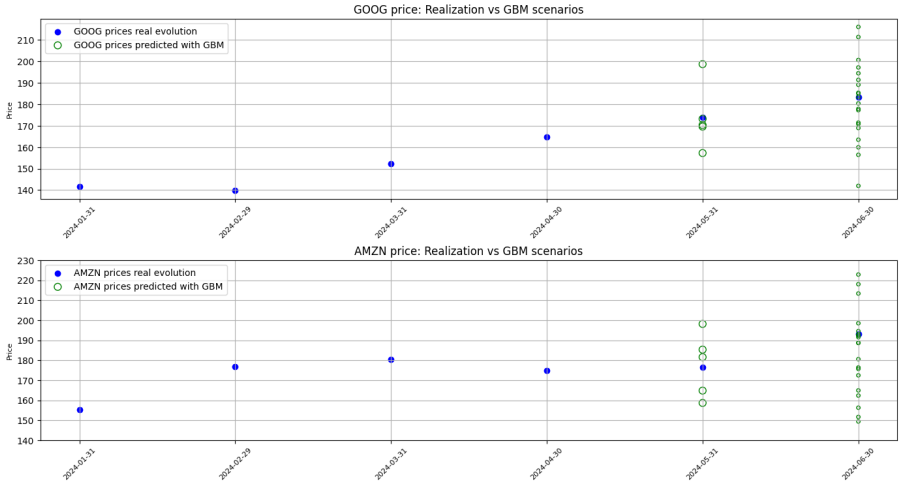

Figure 5.8: Third case: GBM scenarios versus real evolution of shares prices (dimension of nodes are proportional to their probability)

It is possible to see that both methods produce prices spread around the real price, without differing too much from it. Prices produced through moment matching are more centered around the real price and scenarios that deviate the most from the real one have lower probabilities. Prices produced through GBM method are more widespread and have all the same probability. In conclusion, it seems reasonable to assert that if the investor has and wants to exploit information about possible stock performance, then the moment matching method seems preferable. Conversely, if she is not too convinced about how to model uncertainty and wants to take more into account possible extreme scenarios, then it is better to use the GBM method.

5.3.2 Model resolution

Node	**Prices**		**Portfolio composition**		
	GOOG	AMZN	GOOG	AMZN	Liquidity
Node 0	164.45	175.00	3.35	2.54	0.00
Node 1	151.12	187.80	6.47	0.00	0.00
Node 2	190.46	191.84	5.85	0.03	0.00
Node 3	169.97	167.87	5.83	0.00	0.00
Node 4	174.39	177.40	3.28	2.61	0.00
Node 5	169.23	189.97	3.35	2.54	0.00

Table 5.8: Third case results: optimal portfolio composition at root node and its children

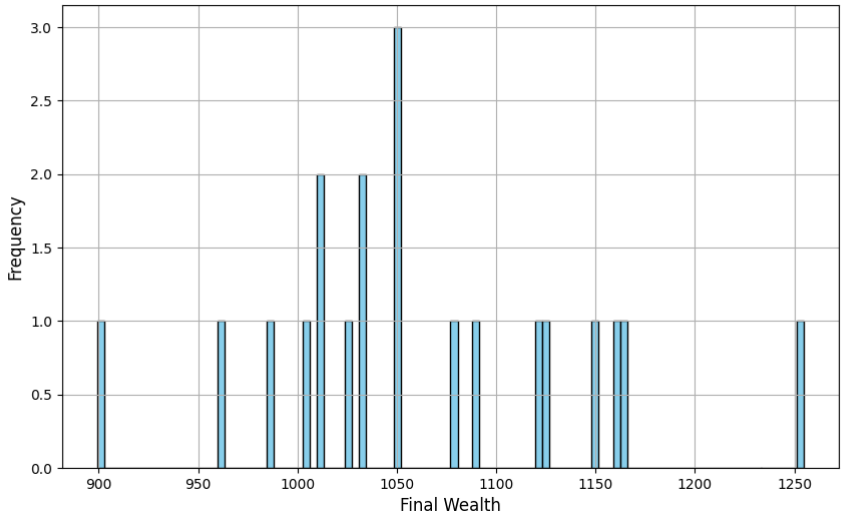

Figure 5.9: Third case results: wealth distribution over leaves of the scenario tree

The tree considered for resolution is the one obtained with the moment matching method (Figure 5.5). The considered portfolio optimization model has the same parameters exhibited in Section 5.1.2. Solving this instance through nested L-shaped approach (as outlined in Section 4.4) means to follow these steps for each iteration:

- At node 0: Solving the master problem and passing the solution (portfolio composition and liquidity) forward to the nodes at $t = 1$;
- For each node $n \in \mathcal{N}_1$: receiving the actions taken at node 0, solving the corresponding subproblem and passing the solution (portfolio composition and liquidity) forward to its own children at $t = 2$;
- For all the leaves: Receiving the actions taken at their parent node, solving the subproblem and passing multipliers backward to their parent node.
- For each node $n \in \mathcal{N}_1$: Receiving multipliers from its children's subproblem and use them to add Benders' cuts as constraints in its own subproblem; solve it and pass backward the multipliers to the root node that will exploit them by adding Benders' cuts to the master problem.

The nested L-shaped algorithm takes 8 iterations to converge at optimal decisions, taking a computational time of 0.97 seconds. Table 5.8 shows the optimal solution, namely, the portfolio composition after first-stage decisions and how it is rebalanced

at the second stage for each possible node. The distribution of the investor wealth over the leaves at $t = 2$ is shown in Figure 5.9. A direct resolution leads to the same results in 0.081 seconds.

5.4 Fourth case: Five assets and three stages

The fourth case is focused on a three-stage setting and a market composed by the risk-free asset and the assets introduced in Section 5.2: XOM, WMT, KO, GOOG and AMZN.

5.4.1 Scenario Generation

The chosen branching factor for this instance was 10 for both the first and the second layer. The scenario tree was built with the MM method, to assess its performance when the number of variables starts to be quite large. In fact, the number of nodes of the tree is 101 and for each node apart the leaves the MM optimization model to solve involved $(5+1)*10 = 60$ decision variables. Generating the entire arbitrage-free scenario tree took a computational time of 166.94 seconds. The squared difference among the expected moments and the produced moments was on average 0,00682. To get a term of comparison, the computational time required to create the tree through Monte-Carlo sampling was 33.71 seconds.

5.4.2 Model Resolution

Node	XOM	WMT	KO	GOOG	AMZN	liquidity
Node 0	8.53	0.00	0.00	0.00	0.00	0.00
Node 1	4.31	8.56	0.00	0.00	0.03	0.00
Node 2	5.76	5.02	0.00	0.00	0.00	0.00
Node 3	4.70	2.87	0.00	0.00	1.16	0.00
Node 4	5.61	5.87	0.00	0.00	0.00	0.00
Node 5	8.41	0.00	0.00	0.00	0.00	0.00
Node 6	3.28	10.27	0.00	0.00	0.00	0.00
Node 7	3.60	6.28	0.00	0.00	1.27	0.00
Node 8	7.04	0.00	0.00	0.00	1.06	0.00
Node 9	2.91	9.37	0.00	0.00	0.54	0.00
Node 10	8.41	0.00	0.00	0.00	0.00	0.00

Table 5.9: Fourth case results: optimal solution

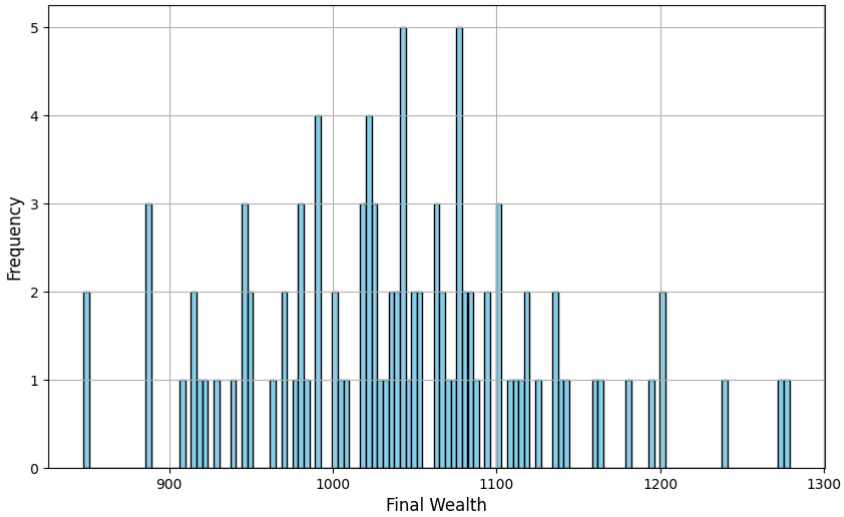

Figure 5.10: Fourth case results: investor wealth distribution over leaves of the scenario tree

The portfolio optimization model had the same parameters of previous instances. The resolution through L-shaped decomposition converged to the optimal solution in 17 iterations, taking a computational time of 16.747 seconds. The optimal solution is sum up in Table 5.9, which specifies the portfolio composition at node 0 and how it is rebalanced according to the realization of uncertainty at the second stage. The distribution of the wealth owned by the investor according to the different leaves at the final stage of the tree is illustrated in Figure 5.10. The same solution was found by direct resolution in 1.48 seconds.

5.5 Stress case

The last instance tested has the goal to show how the L-shaped approach works for cases with higher dimension. In particular, the market considered for this instance was composed by the risk-free asset and 25 assets from the S&P500 index. Three stages were considered, with 30 as branching factor for the root node and the nodes of the second layer. Hence, the total number of nodes in the scenario tree was 931, split in 900 leaves, 30 nodes at $t = 1$ and the root node. The direct resolution by `Gurobi` returned an optimal solution in 25.43 seconds, which leaded to an expected utility of 172.11. For the L-shaped algorithm a time limit of 1500 seconds was set, hence it was stopped after 8 iterations without reaching convergence.

The Figure 5.11 shows the objective value evolution for each iteration of the L-shaped algorithm in order to investigate the rate of convergence. The algorithm

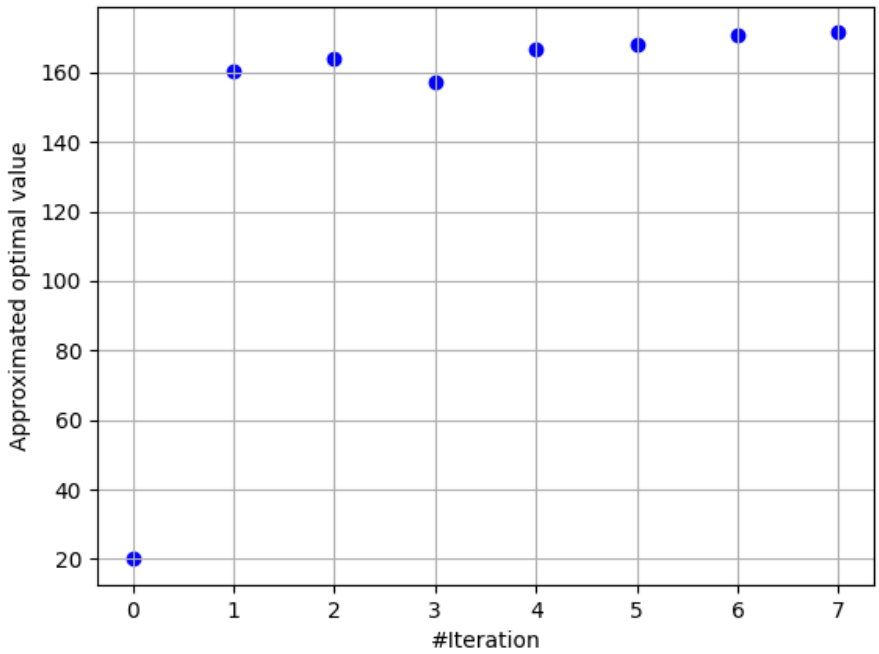

Figure 5.11: Stress case results: L-shaped algorithm rate of convergence

demonstrates rapid improvement in solution quality within just a few iterations, after which its convergence towards the optimal solution slows down. Moreover, this implementation ensures that the algorithm visits the entire tree as early as the first iteration. This results in a great efficiency already from the first iterations. In fact, the second iteration is completed after just 23.96 seconds and yielding a sub-optimal solution with an expected utility of 160.37. This observation suggests that while this implementation of the L-shaped algorithm may not outperform direct resolution in terms of overall speed, it may become highly practical for larger problem instances, especially as the number of stages increases, where direct resolution may also requires significant computation time. In those cases, the L-shaped algorithm can be terminated earlier, once a satisfactory expected utility threshold has been reached, offering a good-quality solution without requiring full convergence.

Chapter 6

Conclusion

This thesis have addressed some of the crucial aspects in the field of stochastic portfolio optimization. Firstly, we have focused on the representation of uncertainty, with an emphasis on what arbitrage opportunities are and how to exclude them while generating the scenario tree. The moment matching method has been presented as a tool to build a scenario tree that is consistent with investor expectations regarding the stock market. As a simpler alternative, we have also illustrated how to generate a scenario tree through Monte-Carlo simulation from a Geometric Brownian Motion. The second topic of this project was the application of L-shaped decomposition to the stochastic portfolio optimization model. Specifically, we have presented this decomposition method through a theoretical discussion regarding its general approach and then it was shown that the form of portfolio optimization model is suitable to be solved with the L-shaped decomposition. Then, we deepened how the L-shaped decomposition tackles the portfolio optimization case using a piecewise linear utility function to express risk aversion while preserving linearity. Finally, to handle multistage problems the nested L-shaped approach was introduced. We have implemented all the methods presented in Python and some computational experiments have been conducted. Concerning the scenario generation, the moment matching method has proven great performance in replicating the expected moments and if the number of assets considered and the children to generate are not extremely large, then the method finds the optimal tree in a fair time. In particular, we pointed out that the moment matching method is suitable especially if the investor has some beliefs regarding future evolution of stock prices or if she wants some peculiar feature to be reproduced in the scenario tree. Otherwise, the sampling from Geometric Brownian Motion is a compelling method, as it is fast and it allows to handle a greater number of scenarios, on the other hand it gives less control to the investor during the generation. Regarding the model resolution, the L-shaped decomposition proved to be an appropriate method to solve the portfolio optimization model in both the two-stage and the three-stage settings. In fact, the

L-shaped decomposition algorithm converged to the optimal solution within a fair computational time in almost all the tested instances, with the exception of the stress case. Even if the method did not outperform the direct solution in terms of computational time in any of the analyzed cases, it still remains a viable alternative for solving stochastic portfolio optimization problems. In fact, on one hand this implementation demonstrated very fast convergence during the initial iterations. This can be especially beneficial as the problem size increases significantly, both in terms of the number of variables and stages, since it gives the possibility to early stop the algorithm when the expected utility exceeds a chosen threshold. On the other hand, improvements to the code or implementation of heuristics, such as sampling the nodes at which to solve subproblems, could improve the efficiency of the algorithm. To sum it up, both the moment matching method and the L-shaped decomposition algorithm deepened in this thesis appear to be effective and efficient in dealing with stochastic portfolio optimization problems.

Appendix A

Code Documentation

All methods covered in the thesis have been implemented in Python. This section aim to give an overview of the code structure.

A.1 Scenario Generation

A.1.1 Class `ScenarioTree`

The class `ScenarioTree` actually builds the scenario tree, using the abstract class `DiGraph` of the library `networkx`. It takes as input:

- `branching_factors`: A vector, whose length is equal to the depth of the tree, that indicates how many children a father has at each stage;
- `n_shares`: An integer to state how many shares are considered in the portfolio;
- `initial_share_prices`: A vector that indicates share prices at node 0;
- `stoch_model`: An instance of the abstract class `StochModel` that will be used to appropriately generate the nodes of the tree.

The builder method `__init__()` calls iteratively the `stoch_model` instance passing to it the number of children to generate (defined by the branching factor) and the parent node, receiving back prices and probabilities that compose the children of the given parent node. Starting from node 0, this process is repeated for every node of the tree apart the leaves (that do not have children). A method `plot` was implemented to give the possibility to display the created scenario tree.

A.1.2 Class `StochModel`

The class `StochModel` is an abstract class, whose instances are the methods used to create the tree. The builder takes as inputs `stoch_setting`, that is a *json* file that contains some settings:

- `tickers`: A vector containing the name of the considered assets, used to extract historical data through the library `yfinance`;
- `start`: The first date of the considered period of time for the collection of historical data;
- `end`: The last date of the considered period of time for the collection of historical data.

The main method of this class is called `simulate_one_time_step`. It takes as input the parent node and the number of children to generate and returns corresponding prices and probabilities to generate the children of the parent node for the scenario tree. `momentMatching` and `geometricBrownianMotion` are the two different instances of this abstract class. Each of them implements the method `simulate_one_time_step` in the way that was explained in Section 3.

A.1.3 Class `multistagePortfolioInstance`

The class `multistagePortfolioInstance` creates an instance of the portfolio optimization model through its builder method `__init__`, which takes as input:

- `inst_setting`: A *json* file that contains some parameters of the portfolio optimization model, such as the risk-free rate, the transaction costs and the initial prices of the stocks;
- `branching_factors`: A vector, whose length is equal to the depth of the tree, that indicates for each stage how many children a father has;
- `plot_tree`: A boolean variable that should be set to True if we want the generated scenario tree to be displayed.

In creating the instance of the problem, this class calls the class `ScenarioTree` passing to it all the necessary parameters to build the tree. After the instance is created, it is passed to the solver for the resolution.

A.2 Model resolution

A.2.1 `Class: Nestedlshaped`

This class implements the algorithm to tackle three-stage portfolio instances. The `__init__()` method takes as input the instance of the problem previously created through the classes explained in the previous Section. The method called `solve(self, max_iter, tol)` is the one that actually implements the loop to solve the problem, taking as input the criteria for the stopping condition. In particular, it iteratively calls the methods:

- `masterpbm(self, multipliers)`: It implements the master problem corresponding to the root node, taking as input the multipliers coming from its children;
- `innerpbm(self, multipliers, Xprev, Wprev, forward)`: It implements the subproblem corresponding to an inner node of the tree. It takes as input:
  - `multipliers`: The optimal dual variables from its children to generate the Benders' cuts;
  - `Xprev`: The portfolio composition found by solving the problem corresponding to its parent node;
  - `Wprev`: The liquidity found by solving the problem corresponding to its parent node;
  - `forward`: A binary input, set to 1 if we are moving forward the tree in the resolution, hence we want the method to return the optimal solution. Otherwise, if set to 0 the method returns the optimal dual variables.
- `leafpbm(self, Xprev, Wprev, Rprev)`: It implements the problem corresponding to a leaf node of the tree. It takes as input the solution found by solving the subproblem of the parent node of the leaf.

Bibliography

[1] C.M. Mesquita, C.A. Valle, and A.C.M. Pereira. «Scenario Generation for Financial Data with a Machine Learning Approach Based on Realized Volatility and Copulas». In: *Computational Economics* 63 (Apr. 2023) (cit. on p. 1).

[2] Paolo Brandimarte. *An Introduction to Financial Markets: A Quantitative Approach*. John Wiley Sons, Hoboken, NJ, 2018 (cit. on pp. 5, 6).

[3] Michal Kaut and Stein W. Wallace. «Evaluation of scenario-generation methods for stochastic programming». In: *Pacific Journal of Optimization* 3 (2003) (cit. on p. 7).

[4] Pieter Klaassen. «Discretized reality and spurious profits in stochastic programming models for asset/liability management». In: *European Journal of Operational Research* 101 (1997), pp. 374–392 (cit. on p. 9).

[5] Alois Geyer, Michael Hanke, and Alex Weissensteiner. «No-arbitrage conditions, scenario trees, and multi-asset financial optimization». In: *European Journal of Operational Research* 206 (Mar. 2010), pp. 609–613 (cit. on pp. 9, 11).

[6] Pieter Klaassen. «Comment on "Generating Scenario Trees for Multistage Decision Problems"». In: *Management Science* 48 (2002), pp. 1512–1516 (cit. on pp. 9, 14).

[7] Alois Geyer, Michael Hanke, and Alex Weissensteiner. «No-arbitrage bounds for financial scenarios». In: *European Journal of Operational Research* 236 (Jan. 2014), pp. 657–663 (cit. on p. 11).

[8] Kjetil Høyland and Stein W. Wallace. «Generating Scenario Trees for Multistage Decision Problems». In: *Management Science* 47(2) (Feb. 2001), pp. 295–307 (cit. on pp. 12, 14).

[9] Alois Geyer, Michael Hanke, and Alex Weissensteiner. «Scenario tree generation and multi-asset financial optimization problems». In: *Operations Research Letters* 41 (June 2013), pp. 494–498 (cit. on p. 13).

[10] J.F. Benders. «Partition procedures for solving mixed-variables programming problems». In: *Numerische Mathematik* 4 (1962), pp. 238–252 (cit. on p. 19).

[11] James K. Murphy. «Benders, Nested Benders and Stochastic Programming: An Intuitive Introduction». In: *ArXiv* abs/1312.3158 (2013) (cit. on pp. 19, 28).

[12] R.M. Van Slyke and Roger Wets. «L-Shaped linear programs with application to optimal control and stochastic programming». In: *SIAM Journal on Applied Mathematics* 17 (1969), pp. 638–663 (cit. on p. 19).

[13] Diana Barro and Elio Canestrelli. «Dynamic portfolio optimization: Time decomposition using the Maximum Principle with a scenario approach». In: *European Journal of Operational Research* 163 (2005), pp. 217–229 (cit. on p. 21).