University of Padova

Department of Mathematics "Tullio Levi-Civita"

Master Thesis in Data Science

Leveraging DAGs for asset allocation

Supervisor

Prof. Alberto Roverato

University of Padova

Co-supervisor

Dott. Bonchi Francesco

Centai Institute S.p.a.

Master Candidate

Lorenzo Caprini

Student ID

1234390

Academic Year

2021-2022

Abstract

The aim of this thesis is to develop optimization techniques for financial portfolios, in order to exploit information regarding causal relationships between considered financial variables, described through directed acyclic graphs (DAGs) encoding the causal structure underlying the data. More precisely we consider: (i) a budget B; (ii) a set of N_A investible financial assets; (iii) a set of N_F non-investible financial factors, causally determining the evolution of the returns for the considered financial assets. The objective of this thesis is to investigate the utility of causal information in asset allocation tasks by theorizing and testing different models for portfolio optimization. These models should be more resistant to sudden shocks to the market structure and should loose less in terms of performance when the system is subjected to an unpredictable shock.

This work is divided into three main chapters: In chapter 2 some theoretical background material about portfolio optimization, DAGs and causality is introduced; in chapter 3 are introduced portfolio optimization models based on Markovitz's framework and afterwards are theorized and explained a series of different methods for asset allocation based on graph clustering techniques; In chapter 4 all the models presented in the previous chapter are tested against a randomly sampled dataset based on the causal structure of the system, both in the static and intervened cases (where a sudden event changed the causal structure), in order to assess their performances before and after a shock occurred and the results obtained are discussed.

Results showed the utility of causal information and causal graph structure in asset allocation tasks and causal models proposed proved to be more stable than benchmark ones in case of soft and hard interventions on the system. Future research can be conducted to improve further the methods proposed and to better exploit causal information and graph clustering techniques.

Acknowledgments

This thesis was made possible thanks to Centai Institute S.p.a., where I spent my internship working on my thesis while supervised by Dott. Bonchi Francesco, Dott. De Francisci Morales Gianmarco and PhD student D'Acunto Gabriele.
Here I found an inspiring working environment, where I was followed closely during the definition and the realization of my project by a team of competent and passionate researchers, who guided me into the research world.

In particular I want to thank Dott. Bonchi Francesco, for giving me the opportunity to work in such a state-of-the-art environment and for his willingness to follow me along my internship.

Also, I want to thank Dott. De Francisci Morales Gianmarco and D'Acunto Gabriele for daily following me in my internship work and for guiding me, with ideas and suggestions, in the realization of my project.

Finally, I want to thank Professor Roverato Alberto for guiding me in the writing of this thesis with precision and willingness and for accepting my project and my ideas with interest.

Contents

Listing of figures

Listing of tables

Listing of acronyms

DAG . . . . . Directed Acyclic Graph

QP . . . . . Quadratic Programming

BN . . . . . Bayesian Network

SCM . . . . . Structured Causal Model

GFT . . . . . Graph Fourier Transformation

MDD . . . . . Maximum Drawdown

1 Introduction

In the last few years, global economy has been shaken to the foundations by a number of unexpected and unpredictable events that impacted greatly on the financial world as a whole. Things such as a global pandemic and the Ukrainian armed conflict hit hard on both big investment funds and small single investors, that were relying on classical financial tools (unable to deal with such extreme cases) to optimize their investments in the form of *assets portfolios*. For this reason, during my internship at Centai Institute S.p.a., we decided to research a way to build a novel method for asset allocation tasks in order to produce portfolios more stable to unexpected and sudden changes to the global market structure.

The scope of this work was to propose a novel approach to asset allocation problems, in order to overcome the instability issues of classical portfolio optimization techniques with the exploitation of two important mathematical concepts: the concept of *causality* and its representation through the use of *Directed Acyclic Graphs* (DAGs). The basic idea behind this study is the fact that methods currently used to tackle these problems are relying just on correlation measures between investible assets A, completely discarding information about the causal structure of the financial universe; if we could correctly analyze this structure it could be possible to better predict failures to specific market nodes thanks to the knowledge of their causes, resulting in more stable portfolio allocations. Therefore, the main focus of the research work was on understanding whether the analysis of the causal structure underlying financial data could bring any advantage to asset allocation tasks, in order to make them more reliable in times of high market distress.

This thesis aims to present ideas and models elaborated to tackle this problem and is divided into three main parts: In chapter 2 some background material about portfolio optimization, graph theory and the mathematical concept of causality is presented, in order to introduce the main theoretical reference to this work. In chapter 3, all the models exploited during this research work are presented in their theoretical form and finally, in chapter 4, the previously presented models are tested against a sampled dataset, simulating their behaviour in situations of stability and under sudden shocks to the market structure.

It is important to remember that this study does not have the claim to produce the best possible method for asset allocation tasks, but it aims to demonstrate that the knowledge of the causal structure underlying the data could be a precious tool to evaluate and produce well balanced and stable financial portfolios that could absorb the effect of sudden changes to the market structure better than traditional methods. Also, we will not cover the problem of *causal inference* and we will assume the causal graph underlying the data as already known.

2

Theoretical Introduction to Portfolio optimization and causal DAGs

As already stated in the introduction, the main idea of this paper is to exploit the notion of causality in Directed Acyclic Graphs (DAGs) to improve the stability of financial portfolios during times of financial distress such as a global pandemic or a drastic fall of the market. To better understand the ideas behind the model proposed in this work we first have to look at some background material that will explain at least in part the theoretical basis behind this thesis. We will first introduce some notes on basic portfolio optimization techniques in section 2.1, then we will move on to introduce graphical representations and some basics of graph theory in section 2.2 and lastly we will introduce the concept of *causality* with its mathematical formalism in section 2.3.

2.1 Portfolio optimization

It is decades that the problem on how to optimize a financial portfolio rises questions among economists and mathematicians, since an easy and efficient way to tackle this task would result in a drastic change in how we invest our money and in the type of risk we incur when doing so. Basically, we are looking for an instrument, in the form of a mathematical equation or an algorithm, that tells us how to invest and how to be a better investor, limiting the risks while

maximizing the return, using the limited amount of historical data on stock prices that we have at disposal. In other words we want to exploit historical data to decide where to spend our money and in what financial assets to invest.

The question that we are posing is this: we have a budget B and a set of N financial assets to invest in (those can be stocks, bonds, real estate investments etc.) how do we divide the total budget B in order to maximize the expected return and reducing at the same time the risk of failure? This is the question that Henry Markovitz, an American economist, tried to address with its work in 1952 when he introduced the *Mean-Variance* framework for portfolio optimization, giving birth to the portfolio allocation practice in finance.
In the same period of time, Markovitz was also working on algorithms to solve quadratic programming problems and exploited its research on this field in order to generate the best possible allocation strategy for an N-assets financial portfolio.
Mean-Variance optimization is still utilized up to these days, for a various number of reasons, first of all the fact that is based on quadratic programming (QP), thing that results in a problem that is easily manageable and solvable; also, in commerce there is a various number of tools that help to solve QP problems in a very straightforward and fast way.
To better understand how the mean-variance optimization works, some basics quadratic programming theory will be introduced first.

2.1.1 Quadratic Programming

The simplest way to describe a Quadratic Programming problem is as an optimization problem with quadratic objective function and linear inequality constraints of the type $Sx \leq T$.

$$\begin{aligned} x^* = \underset{x}{\operatorname{argmin}} &\left(\frac{1}{2}x^T Q x - x^T R\right). \\ s.t. \quad & Sx \leq T \end{aligned} \tag{2.1}$$

where x and R are $N \times 1$ vectors and Q is a $N \times N$ matrix [1]. This formulation allows to set up also equality constraints ($Ax = B$) or box constraints ($x^- \leq x \leq x^+$) alongside the inequality ones in the form

$$\begin{cases} Ax = B \\ Cx = D \\ x^- \leq x \leq x^+ \end{cases} \tag{2.2}$$

that can be translated into linear inequality ones with this system of equations.

$$\begin{bmatrix} -A \\ A \\ C \\ -I_n \\ I_n \end{bmatrix} x \leq \begin{bmatrix} -B \\ B \\ D \\ -x^- \\ x^+ \end{bmatrix}. \tag{2.3}$$

If Q is a symmetric positive definite matrix the solution exists since the function $f(x)$ is convex [1].

2.1.2 Mean-Variance optimization

In his work, as previously said, Markovitz defined precisely what does *portfolio selection* means [1]:

> *"The investor does (or should) consider expected return a desirable thing and variance of return an undesirable thing".*

Given this quote, Markovitz defined the best portfolio allocation as the one that maximizes the expected return for a given level of risk and minimizes at the same time the risk for a desired level of expected return [1]. We consider N assets and we need to find the vector $w = (w_1, w_2, ...w_N)$ of the weights of the portfolio that defines what percentage of budget should allocated to each asset considered. It is also needed to fully invest the portfolio, meaning that the sum over all the weights $w_1, ...w_N$ should be equal to one

$$\sum_{i=1}^{n} x_i = 1_n^T x = 1 \tag{2.4}$$

and we define $\mathbf{R} = (R_1, R_2, ...R_N)$, representing the vector of asset returns, where R_i is the return of asset i and the return of the whole portfolio is given by

$$R(x) = \sum_{i=1}^{n} x_i R_i = x^T \mathbf{R}. \tag{2.5}$$

The objective of this asset allocation problem is, as already said, to maximize the expected return while minimizing the risk (represented by the volatility of the portfolio), so we need to define

the portfolio's expected return

$$\mu(x) = \mathbb{E}\big[R(x)\big] = x^T\mu \tag{2.6}$$

and the portfolio variance is equal to

$$\sigma^2(x) = \mathbb{E}\Big[\big(R(x) - \mu(x)\big)\big(R(x) - \mu(x)\big)^T\Big] = x^T\Sigma x \tag{2.7}$$

where $\Sigma = \mathbb{E}\Big[(\mathbf{R} - \mu)(\mathbf{R} - \mu)^T\Big]$ is covariance matrix of expected returns and $\mu = \mathbb{E}\big[\mathbf{R}\big]$ is the vector of expected returns [1].
Returning on Markovitz's work, the idea is to solve a problem that:

1. maximizes the expected return of the portfolio under a volatility constraint

$$\max \mu(x) \quad s.t. \quad \sigma(x) \leq \sigma^* \tag{2.8}$$

2. minimizes the volatility under an expected return constraint

$$\min \sigma(x) \quad s.t. \quad \mu(x) \geq \mu^*. \tag{2.9}$$

The reduction of these two sub-problems into a single one is needed, since we have to extract just a single weight vector w to tell us how to allocate the budget B, and for this reason a new formulation is needed, one that enables us to maximize the expected return and minimize the volatility at the same time. What Markovitz proposed in his paper was a problem of the form

$$\mathcal{U}(x) = x^T\mu - \frac{\varphi}{2}x^T\Sigma x \tag{2.10}$$

where the parameter φ is called *risk aversion parameter* and represents the tendency of the investor to avoid high risk investments [1]. Since we are better at solving minimization problems than maximization ones, instead of maximizing $\mathcal{U}(x)$ is preferred to minimize $-\mathcal{U}(x)$. In this way Markovitz's problem (2.10) can bi viewed as a Quadratic programming problem:

$$\begin{aligned} x^*(\gamma) = \underset{x}{\operatorname{argmin}}&\Big(\frac{1}{2}x^T\Sigma x - \gamma x^T\mu\Big) \\ s.t. \quad & 1_n^T x = 1 \end{aligned} \tag{2.11}$$

where $\gamma = \varphi - 1$.

We are effectively considering a QP problem where $Q = \Sigma$, $R = \gamma\mu$, $A = 1_n^T$ and $B = 1$. In this way we have a standard and tractable problem for witch it is easy to set up bonds on weights, inequality between asset classes etc [1].

From this point, since the original problem was reduced to a QP problem, we can exploit methods such as the *interior points method* or any form of *gradient projection method* used commonly to solve QP problems. These methods are widely available and are integrated in the majority of mathematical software and in most of the mathematical programming languages such as Python, MATLAB and R.

2.1.3 Risk-balanced portfolios

Markovitz Portfolios, as seen in section 2.1.2, work on quadratic programming in order to maximize the expected return while minimizing the overall risk of the full invested portfolio. This allocation technique is fast and provides good results, but is too sensitive even to the smallest change in asset values, and so it is normally regarded as a risky technique when considering to invest in assets with high volatility. *Risk-Parity allocated portfolios*, another family of portfolio allocation techniques, tries to solve this issue focusing more on estimating and balancing risk rather than maximizing return. The idea is to construct a *well diversified portfolios* by distributing the overall risk equally across different clusters of assets in a *graph-structured portfolio*. We will later see this type of allocation strategies when we will work on *minimum portfolio cuts* and *graph spectral clustering techniques*.

2.2 Graph theory notes

In many nowadays problems involving multivariate modeling, even the simplest tasks can become extremely costly in term of computational time, due to the huge number of variables considered in real-world scenarios. For this reason Graph theory is widely used to treat problems such as optimization ones. A graph is a powerful tool that can encode information about variables in order to formalize the probabilistic and causal structures underlying the data at our disposal, facilitating computations and utilizing information that would not be considered just using raw data. In the next paragraphs will be introduced basic notions and concept regarding graph theory, later to be used to model our data in order to give them a causal interpretation.

2.2.1 Basic definitions

We can define a Graph $\mathcal{G} = \{X, \mathcal{E}\}$ as a collection of nodes $X = \{X_1, ...X_N\}$ and edges $\mathcal{E} \subset N \times N$ ([2][3]). In this case, each node X_i with $i = 1, ..., N$ (where N is the number of nodes in the graph) represents a different financial asset to be invested in, whereas each edge that connects two nodes (and it is denoted as a pair $(i, j) \in \mathcal{E}$, where (i, j) represents the edge connecting nodes i and j) represents a particular relation between the two assets considered.
Any two nodes connected by an edge are considered *adjacent*. Edges can be *directed*, for which we write $(i \rightarrow j)$ or $(j \rightarrow i)$ to indicate the direction or *undirected*, for which we don't define a direction, and we write $(i - j)$ [2].
There is an *undirected path* from i to k if exists a sequence of distinct nodes $X_i....X_k$ connected to each other regardless the type of connection they have (directed or undirected).
There is instead a *directed path* if in the path from X_i to X_k the direction of the arrowheads is consistent (always pointing in the same direction).
A *cycle* occurs when there exists a path from a node X_i to itself of the type $(X_i....X_k, X_i)$ and it can be directed or undirected, depending on the type of edges in the path.
We can talk about a *full connectivity* when there is a connection between every pair of nodes in the graph, so when $(X_i, X_j) \in \mathcal{E} \quad \forall \quad X_i, X_j \in \mathbf{X}$ with $\mathbf{X}$ the set of all nodes in the graph.

2.2.2 DAGs and Bayesian Networks

When a graph $\mathcal{G}$ has just direct edges and it doesn't show any cycle we can talk about a *Directed Acyclic Graph* (DAG). In a DAG we can define a *parent* X_i of the node X_j if $(i, j) \in \mathcal{E}$ but $(j, i) \notin \mathcal{E}$ (if there is a direct connection from X_i to X_j) and in this case X_j is called *child* node to X_i. If there is a path from X_i to X_j, X_j is called a *descendant* of X_i and X_i is called an *ancestor* for X_j. If no paths are present between two nodes, one is simply called a *non-descendant* of the other node and vice versa [2].

Defined what are the relationships between particular nodes in a directed acyclic graph, we can briefly describe some interesting structures that can be systematically encountered when dealing with graphs. In a DAG we have a *v-structure* when two non-adjacent nodes are parents of the same child node, called in this case a *collider*

$$(X_j \rightarrow C \leftarrow X_i). \tag{2.12}$$

Also, the exact opposite of a collider can be defined when it exists a node that is a parent for other

two non-adjacent nodes (with no connection between them), we call this node a *confounder* [3].

$$\big(X_j \leftarrow H \rightarrow X_i\big). \tag{2.13}$$

To transpose graph information in a more mathematically tractable representation we can use the matrix associated with the DAG that for simplicity will be called *weight matrix*. What was called a weight matrix is a structure that encodes the connections of a graph in a matricial form, where each row vector represents a variable X_i and each value of that vector represent the interaction of node i with other nodes; interaction can be positive or negative, whereas a value of 0 means that there are no interaction between node X_i and X_j with $j = 1, .., i-1, i+1, ...N$. In other words, each value (i, j) of the weight matrix defines if there is a parental relationship between X_i and X_j.
For example, if we consider this simple system of equations representing a direct graph

$$\begin{aligned} X_1 &= 2X_2 + 3X_3 \\ X_2 &= 3X_3 + 2 \\ X_3 &= 2X_1 + 1 \end{aligned} \tag{2.14}$$

we can write its weight matrix as

$$\begin{bmatrix} 0 & 2 & 3 \\ 0 & 0 & 3 \\ 2 & 0 & 0 \end{bmatrix}. \tag{2.15}$$

Once defined how the graph is represented, in order to give a real world meaning to this kind of structure we need to associate a probability distribution to each one of the nodes in the graph. We can so define a *Bayesian Network structure* as a DAG where the nodes represent random variables $X_i....X_N$ and where each variable is conditionally independent from all its non-descendants given the set of its parents [2], so basically we can define a joint probability distribution for the Bayesian Network as

$$P(\mathbf{X}) = \prod_{i=1}^{d} P\big(X_i | pa_i\big) \tag{2.16}$$

where with pa_i we indicate the set of the parents of node i [3].
Furthermore, we can say that $P\big(X\big)$, as the joint probability distribution of the graph, factorizes over $\mathcal{G}$ and we can more correctly define a Bayesian Network as the couple $(\mathcal{G}, P)$ where P

factorizes over $\mathcal{G}$.

2.2.3 The Markov Property

Markov property is a property that helps to understand the relation between graphs and probability distributions and it is normally assumed as true when dealing with graphical models, especially in the field of causal inference. The exact definition of Markov property is given below:

> *Given a DAG $\mathcal{G}$ and a joint distribution P_X, this distribution is said to satisfy:*
>
> 1. *the **global Markov property** if*
>
> $$\mathbf{A} \perp\!\!\!\perp_{\mathcal{G}} B \mid \mathbf{C} \Rightarrow \mathbf{A} \perp\!\!\!\perp B \mid C \tag{2.17}$$
>
> *for all the disjoint vertex sets A, B, C (where symbol $\perp\!\!\!\perp_{\mathcal{G}}$ denotes d-separation, defined below)*
>
> 2. *the **local Markov property** if each variable is independent of its non-descendants given its parents*
> 3. *the **Markov factorization property** with respect to DAG $\mathcal{G}$ if*
>
> $$p(\mathbf{x}) = p(x_1, \ldots, x_d) = \prod_{i=1}^{d} p(x_i | pa_i^{\mathcal{G}}) \tag{2.18}$$
>
> *with $p(\mathbf{x})$ density of distribution $P(X)$ and $pa_i^{\mathcal{G}}$ as the parents of node i in graph $\mathcal{G}$ [4].*

Basically, as introduced before when talking about Bayesian Networks, one can predict the value of a specific node X_j just looking at the variables that are not conditionally independent from it. This helps to reduce the complexity of the computations, especially in large networks, and can be formalized with the concept of *Markov blanket*.

The Markov Blanket of a node X_j in a DAG $\mathcal{G} = \{X, \mathcal{E}\}$ [4] is the smallest set M such that

$$Y \perp\!\!\!\perp V \backslash (\{Y\} \cup M) \quad given \quad M \tag{2.19}$$

that basically means that information about node X_j cannot be extracted from nodes outside of its Markov blanket M.

2.2.4 D-SEPARATION, D-FAITHFULNESS AND SUFFICIENCY

Strictly related to the previously defined Markov Property lies the notion of *d-separation*. In a DAG $\mathcal{G}$ we can define that a path between nodes i and j is blocked by a set S whenever there is a node k such that $k \in S$ in one of these three cases [3]:

$$\begin{aligned} &i \to k \to j \\ &i \leftarrow k \leftarrow j \\ &i \leftarrow k \to j \end{aligned} \tag{2.20}$$

or such that k and no descendants of k are in S and

$$i \to k \leftarrow j. \tag{2.21}$$

The definition of d-separation implies that if i and j are d-separated by node k, if we observe the state of k, evidence propagation between i and j (or j and i) is blocked and so it's possible to say that d-separation implies conditional Independence.

Next, we can report here the definition of *d-faithfulness*:

> *A distribution is d-faithful to a DAG $\mathcal{G}$ if no conditional independence relations other than the ones entailed by the Markov property are present [5].*

Basically this translates into the fact that all the conditional independence relations are exclusively the ones represented in the DAG.

Lastly we can talk about another assumption usually considered when handling Bayesian Networks i.e. *sufficiency assumption* [3].

This assumption is related to DAGs derived from observational data, where it is important to be sure to have collected enough data to define a correct representation of the graph structure. Sufficiency of the graph is assumed when there are no unobserved confounders that can lead to a wrong interpretation of the system, given that the absence of a confounder could be confused with non existing edges between nodes that normally would have been d-separated, giving birth to a wrong joint probability distribution for the BN.

2.2.5 The problem with causal interpretation of DAGs

To complete this introduction to graph theory we want to talk briefly about the interpretation of a DAG such as a structure representing real world data. This might seem an obvious thing, but the existence of direct links in DAGs not always reflect the presence of a cause-effect relationship between two variables i and j [2]. A direct link between these two variables could in fact indicate a cause-effect relation between parent and children, but it could also just signal the presence of a strong correlation, in the case for example of a hidden unobserved node that works as a confounder, and for this reason when building a Bayesian Network it is good practice to know exactly the cause-effect relationships present in the data.

From now on, the models we will work on will be considered as DAGs respecting the previously introduced sufficiency assumption, meaning that no hidden confounders are present and, having imposed the underlying cause-effect structure of the data,our DAGs will represent causal relations between nodes.

2.3 Causality notes

As stated in the introduction of this work, the fundamental idea of this thesis is to exploit the knowledge of the causal structure behind financial data in order to improve the efficiency of portfolio allocation strategies. To tackle this problem we have to introduce formally the concept of *Causality*. The study of causality is simply the study of cause-effect relations between mathematical entities in the form of random variables. The first thing we need to know when dealing with causality is that a causal structure, that defines the cause-effect relations between variables, entails not just a probability distribution, but also other things, such as an interventional distribution and counterfactual reasoning. This makes causal reasoning much more complete and powerful than probabilistic reasoning, since we are considering a richer and more complex model underlying the data, that allows the study the effect of interventions and other modification to the structure of the model considered.

Basically there are more information to use and this could lead to a greater comprehension of the phenomena we are trying to explain using causal models.

2.3.1 Structural Causal Models

In the previous paragraph we briefly introduced what is causal reasoning, why it is worth considering it and why it can be good in certain situations, but now we try to introduce formally a way to represent it mathematically.

We will start from a basic definition of a *Structural Causal Model*:

> *A **Structural Causal Model (SCM)** $\mathfrak{C} := (S, P_N)$ consists of a collection S of n (structural) assignments of the type*
>
> $$X_i := f_i(pa_i, N_i) \quad i = 1, ..., n \tag{2.22}$$
>
> *where $pa_i \subseteq C\{X_1, ..., X_n\}\backslash\{Xi\}$ are the parents of X_i; and a joint distribution $P_{\mathbf{N}} = P_{N_1}, ..., P_{N_n}$ over the noise variables, which we require to be jointly independent, for which $P_{\mathbf{N}}$ is a product distribution. The graph $\mathcal{G}$ of an SCM is obtained by creating one vertex for each X_i and drawing directed edges from each parent pa_i to X_i. We also assume this graph to be acyclic. We sometimes call the elements of pa_i not only parents, but also direct causes of X_i, and we call X_i a direct effect of each of its direct causes [4].*

In practice, an SCM results as a system of equations, where each node is expressed in function of its direct causes (its parents) and a noise factor N_i (usually as a Gaussian white noise, but the distribution can change), exactly as the representation of a Bayesian network defined in section 2.2.2. With this parallelism, we can find the joint distribution that the SCM $\mathfrak{C}$ defines over its variables and we call it the *entailed probability distribution* $P^{\mathfrak{C}}_{\mathbf{X}}$ of the SCM [4].

With this knowledge on how we can define cause-effect relations among variables of our system, we now can exploit theorems, definitions and properties of elements of graph theory to work on causal systems for which we know the SCM and its entailed distribution $P^{\mathfrak{C}}_{\mathbf{X}}$.

Finally, we can assume we are working in a state of *minimality* of the SCM when we are sure that the representation we chose for the model is the simplest possible.

2.3.2 Interventions

Previously we saw how a SCM entails a probabilistic distribution, but as we already said at the beginning of paragraph 2.3, we have much more than that in a causal model. Interventions are mathematical tools that we can use to study how the system behaves when an external agent acts on a previously well determined system [4]. In practice, to intervene on a variable in a system means to directly change its value or distribution, with no effect of parent or ancestor nodes. So, for example, if in a graph we change the distribution of the node X_i to a Bayesian one with $p = 0.5$ (basically a coin flip), this random variable will no longer have any parental direct causal effect, it will depend only on the coin flip.

In this way we effectively changed the causal structure of the graph, since direct edges representing a parental relationship with X_i are no longer present and basically X_i became a source node (or an isolated one it it has no children too).

Using the formalism defined before, we consider an SCM $\mathfrak{C} := (S, P_N)$ and its entailed distribution $P_{\mathbf{X}}^{\mathfrak{C}}$. At this point we can replace previously defined assignments to

$$X_i := \tilde{f}(\tilde{pa}_i, \tilde{N}_i) \tag{2.23}$$

and we find the new entailed distribution that we call the *intervention distribution*; we denote it as

$$P_{\mathbf{X}}^{\tilde{\mathfrak{C}}} = P_{\mathbf{X}}^{\mathfrak{C};do\left(X_i := \tilde{f}(\tilde{pa}_i, \tilde{N}_i)\right)}. \tag{2.24}$$

Here we utilize a particular operator, called the "$do()$" operator [6]. It is used to represent the effect of an external modification, on one or more assignments, to the system as a whole and it differs very much from a normal observation from where we can define a conditional probability. In conditional probability in fact, we ask what is the probability of observing a particular value for a certain random variable given that we know another variable assumed a particular value ($P(A \mid B = 0)$). In this case we observe a realization of variable B and we conserve the underlying causal structure intact. Instead, if we use the $do()$ operator we obtain $P(A \mid do(B = 0))$ where we are basically saying that the value of B was forced to 0 and the causal structure of the model was actively changed into a new one, the intervention distribution.

There are mainly two different types of interventions possible on a DAG [4]:

1. **hard interventions**: interventions that completely change the structure of the graph, for which $\tilde{pa}_i \neq pa_i$, for example assigning a constant value to a specific node, deleting all the parental relationships of the original graph for node i (this in particular is a so called atomic intervention of the type $do(X_i := k$ with k constant)
2. **soft interventions**: also called *imperfect interventions*, where no modifications are made to direct causal interactions between variables and $\tilde{pa}_i = pa_i$ (we can obtain an intervention such as this one by changing for example the noise distribution of a specific variable).

Finally, it is possible to define that exists a *total causal effect* of the variable X_j on

X_i if and only if

$$X \not\perp\!\!\!\perp Y \quad in \quad P_{\mathbf{X}}^{\mathfrak{C};do(X:=\tilde{N}_X)} \tag{2.25}$$

with $\tilde{N}_X$ a random variable of any type [4].

In practice we are looking at the effect of the intervention on other variables that could be effected. If this happens, it means that X_j has a causal effect on that specific variable for which there is a direct parental link from X_j. Obviously there cannot be a total causal effect of X_j on X_i if there is no direct path between the two variables, but sometimes even a direct path is not a sufficient condition for the existence of a total causal effect. The study of Total causal effects is crucial in causal discovery tasks, since we can infer the presence of direct links from a variable to another just by studying the total causal effect obtained after an intervention on the system.

2.3.3 Counterfactuals

At the beginning of section 2.3 we assessed how interventions and observations of the same systems are a radically different thing. Now we try to formalize this thing in the form of counterfactuals. Counterfactual reasoning can be reduced to a specific way of asking *"what would have happened if I would have done that instead of this?"*. We are combining observations of the model and the analysis of the intervention distribution in which all the environment features remain unchanged except for the variable we are considering in the counterfactual statement. Basically we are asking *"what would have been the resulting distribution if i would have done a specific type of intervention on the system?"*.

Formally we can define a counterfactual SCM by changing the structure of its entailed probability distribution using the $do()$ operator to perform an intervention on it

$$\mathfrak{C}_{\mathbf{X}=\mathbf{x}} := \left(\mathbf{S}, P_{\mathbf{X}}^{\mathfrak{C}|\mathbf{X}=\mathbf{x}}\right). \tag{2.26}$$

Combined with interventions, counterfactuals SCMs are one of the main tools for causal structure discovery tasks.

3

Presentation of different theoretical models

In chapter 2 were introduced the theoretical basis to understand the models that were used to work on asset allocation tasks while trying to exploit the causal structure of the data at our disposal. As already stated in the introduction paragraph, we will just work on the task of asset allocation, completely skipping the problem of causal discovery and causal inference of the graph, assuming to already know the complete structure of the SCM $\mathfrak{C}$ and of its entailed probability distribution $P_{\mathbf{X}}^{\mathfrak{C}}$. Knowing this, it is possible to make a few assumptions on the data structure we are going to work on:

- the causal structure of the data respects the Markov property in section 2.2.3, which essentially means that each node only depends from its parents.
- the SCM $\mathfrak{C}$ respect the structural minimality condition
- we can assume the *sufficiency* of the model, that is we take as granted the fact that there are no hidden confounders

Also, since we are dealing with a regular structure, it is natural to assume valid all the properties introduced in chapter 2 about DAGs, Bayesian Networks and SCMs.

In this chapter we will only introduce the models theoretically, we will test them and report the results obtained in chapter 4.

3.1 Sampling data from a theoretical distribution

Before describing the models, it is important to define what type of data we are going to work on and how the sampling of these data took place. As already said in the introduction to this chapter, we will not focus on the problem of causal inference and both the structure and the coefficients of the weight matrix of the causal graph defined by the SCM $\mathfrak{C}$ will be assumed as already known.

The Universe we are trying to model consists of N_a investible assets $A_1, ..., A_{N_a}$, with no direct interaction between them (direct connections between assets are not permitted), and N_f non-investible factors $F_1, ..., F_{N_f}$ that work as parents for the assets A and that can interact between each other (direct connections between factors are permitted). From this point, two separate routes were used to generate the structural causal model:

1. A *manual* generation, where variables were defined one by one
2. A random generation starting from the weight matrix of the causal DAG

3.1.1 Manual definition of the SCM

Following this path, we proceed to define all the variables, starting from random noises as normal distributions with mean μ and variance σ^2

$$Z_1, ..., Z_{N_a+N_f} \sim \mathcal{N}(\mu, \sigma^2) \tag{3.1}$$

then defining the N_f factors (where $a_{i,j} \in \mathbb{R}$ and $a_{i,j} \neq 0 \iff j \in pa_i$)

$$\begin{aligned} F_1 &= a_{1,2}F_2 + a_{1,3}F_3 + \cdots + a_{1,N_f}F_{N_f} + Z_{F_1} \\ &\vdots \\ F_{N_f} &= a_{N_f,1}F_1 + a_{N_f,2}F_2 + \cdots + a_{N_f,N_f-1}F_{N_f-1} + Z_{F_{N_f}} \end{aligned} \tag{3.2}$$

and finally the N_a assets (where $b_{i,j} \in \mathbb{R}$ and $b_{i,j} \neq 0 \iff j \in pa_i$)

$$\begin{aligned} A_1 &= b_{1,1}F_1 + b_{1,2}F_2 + \cdots + b_{1,N_f}F_{N_f} + Z_{A_1} \\ &\vdots \\ A_{N_a} &= b_{N_a,1}F_1 + b_{N_a,2}F_2 + \cdots + b_{N_a,N_f}F_{N_f} + Z_{A_{N_a}}. \end{aligned} \tag{3.3}$$

In manually defining this structure we need to consider what is called the *causal ordering* of the SCM, that is an internal ordering where the variable at $i-th$ position can only be causally influenced by variables in position $\{1, 2, ..., i-1\}$. This will pose some boundaries on how to build the *weight matrix* for the causal DAG related to the newly defined SCM, that must be a permutation of a strictly upper/lower triangular matrix depending on the type of ordering considered (directed or inverse).
From now on the causal ordering considered will be inverse, in other words we will construct the weight matrix with in first position the last element in the causal ordering proceeding till the last row, representing the first element in the causal ordering, not being influenced by any other node.

Knowing this, it is possible to define a matrix C as the weigh matrix of the causal graph associated to the SCM, where the first N_a rows and columns represent ordered assets A and the rest of the rows and columns represent ordered factors F (all factors are higher in the causal ordering with respect to assets and for this reason appear in the last positions of the matrix). Each element of row $i = 1, ..., N_a + N_f$ represent the parental interaction of node i with node $j = 1, ..., i-1, i+1, ..., N_a + N_f$ with element in position (i, j) being different from 0 if and only if $j \in pa_i$. Given how we defined matrix C we can view it as a block matrix

$$C = \begin{bmatrix} \emptyset & B \\ \emptyset & A \end{bmatrix} \tag{3.4}$$

with B of dimension $N_a \times N_f$ (parental interaction of factors with assets)

$$B = \begin{bmatrix} b_{11} & b_{12} & \cdots & b_{1N_f} \\ b_{21} & b_{22} & \cdots & b_{2N_f} \\ \vdots & \vdots & \ddots & \vdots \\ b_{N_a1} & b_{N_a2} & \cdots & b_{N_aN_f} \end{bmatrix} \tag{3.5}$$

and A of dimension $N_f \times N_f$ (parental interactions of factors with other factors)

$$A = \begin{bmatrix} 0 & a_{12} & \cdots & a_{1N_f} \\ 0 & 0 & \cdots & a_{2N_f} \\ \vdots & \vdots & \ddots & \vdots \\ 0 & 0 & \cdots & 0 \end{bmatrix} \tag{3.6}$$

with the first block being 0 due to the fact that no interactions are possible between assets.

3.1.2 Random generation of the SCM

Another method implemented tries to randomly generate both connections and parameters of the causal graph, following the directives given in paragraph 3.1.1.

To do this, we first randomly generate the elements of an empty matrix C (of dimensions $(N_a + N_f) \times (N_a + N_f)$) using as chosen distribution a random normal $\sim \mathcal{N}(\mu, \sigma^2)$. Once this is done, we select random elements of this matrix to eliminate with the aid of a binomial distribution with $p = \frac{1}{2}$, so basically we choose what elements to keep with $(N_a + N_f) \times (N_a + N_f)$ coin flips. We define a matrix B with each element $\in \{0, 1\}$ and we filter matrix C performing a point-wise multiplication with binomial matrix B. Finally, with the aid of the linear algebra package of **numpy**, filtered matrix C is converted into an upper triangular matrix (since there is an inverse causal ordering) and its first block of dimensions $N_a \times N_a$ is set to 0 in order to impose no interactions between different assets. Noise terms are defined exactly as before, one for each assignment, both for assets and factors alike.

Now that we introduced these two different methods, it is possible to go on with the actual sampling of data points from the newly created SCM.

3.1.3 Sampling the data

In both of the previously defined paths for generating a sample SCM the final data structure obtained was the weight matrix C of the causal graph entailed by the SCM. Our goal is to generate a dataset with $N_a + N_f$ columns, where each column represents a different asset or factor and contains a time-series of the returns for that specific node given a determined period of time T expressed as days in a variable number of *trading years* (each trading year has exactly 252 trading days).

Starting from the matricial representation of the SCM we can write

$$\mathbf{X} = C\mathbf{X} + \mathbf{Z} \tag{3.7}$$

where $\mathbf{X}$ is a matrix with $N_a + N_f$ columns and T rows, C is the weight matrix of the SCM $\mathfrak{C}$ and $\mathbf{Z}$ is a matrix of dimension $T \times (N_a + N_f)$ (exactly as $\mathbf{X}$) accounting for the daily error terms for each one of the variables.

From this formulation we can write

$$\begin{aligned} I\mathbf{X} - C\mathbf{X} &= \mathbf{Z} \\ (I - C)\mathbf{X} &= \mathbf{Z} \\ \mathbf{X} &= (I - C)^{-1}\mathbf{Z} \\ \mathbf{X} &= M\mathbf{Z} \end{aligned} \tag{3.8}$$

where $M = (I - C)^{-1}$ and matrix $(I - C)$ it is surely invertible since C it is strictly upper triangular by construction. Knowing matrix C from previous SCM definition we can immediately define M and just the sampling of $\mathbf{Z}$ is needed in order to find dataset $\mathbf{X}$ of daily returns.

Normally, when talking about financial assets (mainly stocks) we are in possession of the *Adjusted Closing Price* from which it is possible to derive the daily return considering the percentage change respect to the previous day's closing price

$$R_{A_i} = \frac{Adj.Close_{A_i}(t) - Adj.Close_{A_i}(t-1)}{Adj.Close_{A_i}(t-1)} \tag{3.9}$$

but since we are allocating portfolio model's based on returns it is better and easier to directly sample these values, since it is possible to take as a general assumption the fact that daily returns usually follow a normal distribution $\mathcal{N}(\mu, \sigma^2)$, where $\mu \to 0$. For this reason, matrix $\mathbf{Z}$ of random noises was sampled as a normal distribution $\mathcal{N}(\mu, 1)$ with a variance of 1 in order not to have excessively volatile assets to work on and $\mu \to 0$ in order to simulate real-world stock returns.

Finally, with a simple matricial multiplication between M and $\mathbf{Z}$ it is easy to obtain the desired matrix $\mathbf{X}$.

3.2 Mean-variance optimization exploiting causal structure of the data

As we introduced in section 2.1.2, mean-variance (or Mean-Risk) optimized portfolios are widely used in finance since their advent in the initial work of Markovitz, given that their formulation is easy to solve thanks to it being in a quadratic form, thing that enables a solution search through one of the many algorithms and tools to optimize QP problems.

Considered that it is so commonly used even nowadays, we decided to take Markovitz's portfolio as one of the baselines to test our models against, in order to see if the

actual knowledge of the causal structure underlying the data really gives an edge over standard and well established method such as Markovitz's one.

To do this, we decided to exploit a Python package called **`Riskfolio-Lib`**, consisting in a wide variety of tools to perform asset allocation tasks such as the optimization of a Markovitz's mean variance portfolio. This tool is based on the classical model formulation described in equations (2.10) and (2.11), deriving the sample mean μ and sample covariance matrix Σ from historical data of the time series provided (the one that was sampled in paragraph 3.1.3). In this case, the optimal portfolio is calculated on the base of the general problem

$$\begin{aligned} \underset{w}{\textit{optimize}} \quad & F(w) \\ \text{s.t.} \quad & Aw \leq B. \\ & \varphi(w) \leq c \end{aligned} \tag{3.10}$$

where $F(w)$ is an objective function chosen by the user, $Aw \leq B$ are the usual linear constraints and $\varphi(w) \leq c$ is a linear constraint on a convex risk measure $\varphi(w)$, also defined by the user [7]. We chose to utilize *Sharpe ratio* as the objective function $F(w)$ and the standard deviation as risk measure $\varphi(w)$.

Sharpe ratio is a function used to calculate the *risk adjusted return ratio* from a portfolio returns series and its definition is reported below:

$$Sharpe(X) = \frac{\mathbb{E}(X) - r_f}{\varphi(X)} \tag{3.11}$$

where X is the vector of portfolio returns (based on portfolio allocation w), r_f is the so called *risk free rate* (usually is 0) and $\varphi(X)$ is the chosen risk measure (in this case the standard deviation calculated on portfolio returns X).

Optimizing the objective function $F(w)$ in this case means maximizing the expected return of the portfolio ($\mathbb{E}(X)$), while minimizing the standard deviation of portfolio returns (in the form of the risk measure $\varphi(X)$) with a risk constraint $\varphi(w) \leq c$.

In this base model, two are the main constraints imposed:

1. The portfolio is fully invested ($\sum_{i=1}^{N_a} w_i = 1$), meaning that all of the budget B at our disposal is allocated onto the different assets considered
2. The portfolio is *long-only* ($w_i \geq 0 \quad \forall i = 1, ..., N_a$), meaning that there is no possibility of *short selling* (all weight values must be greater than 0)

Given all this, we obtain a vector $w_{opt} = \{w_1, w_2, ..., w_{N_a}\}$, where each w_i represents the percentage of budget to be allocated on asset A_i.

As we can see, this type of portfolio does not consider at all the presence of financial factors $F_1, ..., F_{N_f}$ and the causal relations presents in the original DAG are discarded in favour of correlation measures between given assets. This means that we obtain a result based only on the values of variance-covariance matrix Σ, that can be hard to estimate correctly when in presence of large portfolios (with a big number of assets N_a) and with a short time window to work on, thing that happens very frequently in real-world trading scenarios, since each trading year has only a limited amount of days (252) and it is not possible to gain a reasonable amount of points just considering a normal amount of years.

Also, in this way there is a great loss of information, since discarding both direction and weight of all the edges in the graph could be misleading and could lead to the elimination of possible connections between assets, not visible in a correlation environment, but important in extreme situations, leading to a portfolio that is stable in a static scenario (such as the one just considered), but very unstable when considering unexpected events.

3.2.1 EXPLOITATION OF CAUSAL STRUCTURE

As stated before, the classical mean-variance portfolio optimization does not consider any type of information other than correlation measures in the form of variances and covariances contained in Σ and obtained through the study of historical data. The first way considered to inject information about the causal structure in an asset allocation task, was to directly work onto the same mean-variance optimization problem, while modifying the parameters considered.

To understand this we have to go back to equation (3.8), where matrix $\mathbf{M}$ was obtained

$$\mathbf{X} = M\mathbf{Z} \qquad with \quad M = (I - C)^{-1}. \tag{3.12}$$

Previously, we defined $\mathbf{Z}$ as the matrix of error terms, where each error term is distributed as a Gaussian white noise with

$$Z_1, ..., Z_{N_a+N_f} = \mathcal{N}(\mu, \sigma^2) \tag{3.13}$$

with $\mu \to 0$ and $\sigma^2 = 1$.

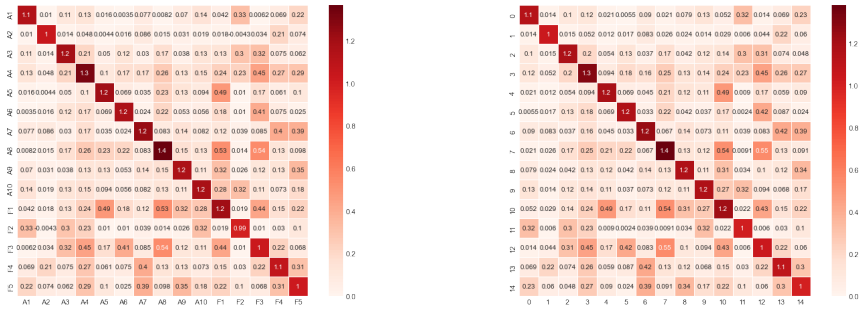
This means that $\mathbf{Z}$ is distributed as a multivariate normal $\mathcal{N}(\mu, \Sigma)$ where μ is equal to the sum of all the single noise factor's means $\sum_{i=1}^{N} \mu_i$ and $\Sigma = \mathbb{I}$. With this

assumptions it is possible to look back at equation (3.12) and notice that, since we consider a static structure for the graph $\mathcal{G}$, M is a constant term and, thanks to the properties of normal distributions, it is possible to say that $\mathbf{X}$ is still a multivariate normal distribution

$$\mathbf{X} \sim \mathcal{N}(\mu_{\mathbf{X}}, M^T \mathbb{I} M) = \mathcal{N}(\mu_{\mathbf{X}}, M^T M) \tag{3.14}$$

where $M^T M$ happens to be the Gram matrix of the vectorial space where the vectors $x_1, x_2, \ldots, x_T$ are the rows of the matrix $\mathbf{X}$. Gram matrix is symmetric and positive semidefinite by construction and thus it is possible to use it in a vast variety of optimization problems.

Until now we have considered an undirected problem where the main focus was on correctly estimating the variance-covariance matrix Σ from the sampled dataset in order to use it for allocating the best portfolio possible, but looking back at equation (3.14) we can see how Σ happens to be simply an estimator for gram matrix G.



(a) Covariance matrix heat-map

(b) gram matrix heat-map

Figure 3.1: representation of the heat-maps for the two different matrices in a system of 24 assets and 12 factors

In fact, as we can see from the plots in figure 3.1, the graphical representation for the two matrices Σ and G it is nearly identical when there is an adequate amount of points to calculate variances and covariances, and this assumption can be demonstrated by computing and visualizing the evolution of the *Frobenious Norm* for the two matrices under exam [8].

Frobenious Norm for an $m \times n$ matrix M is the square root of the sum of the square value of its elements

$$\|M\|_F = \sqrt{\sum_{i=1}^{m} \sum_{j=1}^{m} m_{ij}^2} \tag{3.15}$$

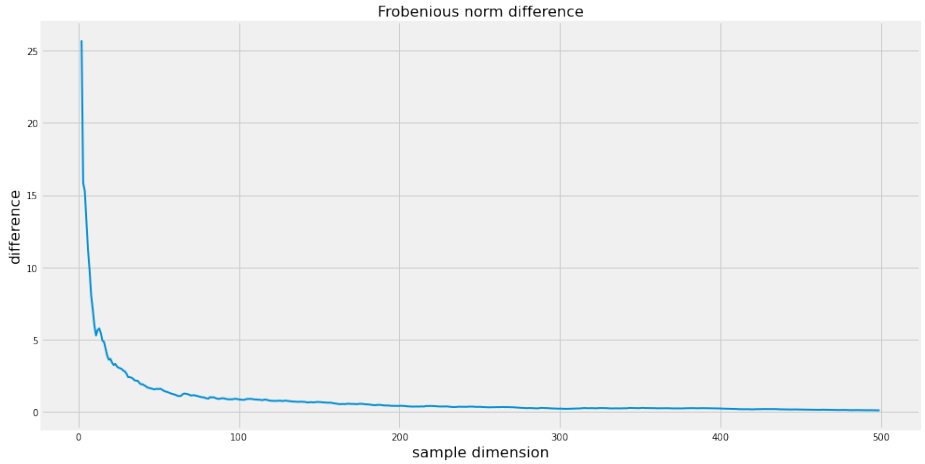


Figure 3.2: Graphical representation of the Frobenious norm difference for an increasing number of data points in a system with 24 assets and 12 factors

and in figure 3.2 it is represented the difference between the Frobenious norm calculate for matrices Σ and G with an increasing number of data points. It is easy to see how the difference tends to 0 when the time-window T increases and for this reason it is safe to assume that Σ is a good estimator of G when there are enough data points to work on.

This means that knowing the causal structure underneath the data should give us an edge even in the classical mean-variance portfolio optimization and it is possible to just reproduce the same optimization process that took place at the beginning of paragraph 2.1.2, with minor changes:

1. Since matrix G is known from the distribution, it is possible to exchange it for Σ in the mean-variance problem (there's no need to estimate G from historical data if the causal structure is already known)

2. Additional constraints of the type $Ax = B$ are needed to eliminate any type of investment on financial factors (the full universe of $A + F$ is considered in gram matrix G, but only assets are investible elements, so it is necessary to impose any possible allocation of factors to 0)

with these adjustments the models should have an edge over the classical version using Σ, at least in scenarios with high values of N_a and low values of T, where the estimation of covariances can result problematic.

3.3 Iterative spectral clustering methods

So far, we worked on methods of mean-variance optimization, where the focus was on maximizing the expected return $\mathbb{E}(X)$ while at the same time reducing the overall risk of the portfolio, expressed as the weighted volatility of the assets.

These methods could maybe produce decent portfolios in a static scenario, but tend to fail when subject to stronger market fluctuations and, since our goal is to exploit causal information to develop an allocation technique that produce more stable portfolios, maybe a complete change of paradigm should occur, to find a method that produces more balanced and diversified allocations.

Also, in the previous methods, to exploit the causal information contained in the data, we just considered the structure of the matrix C in order to derive the Gramian G used to perform mean-variance optimization. This obviously could lead to an advance over standard methods in a situation of short time windows and very wide portfolios, but maybe working directly on the DAG structure could lead to an even greater exploitation of causal information underneath the available data.

For this reason, the primary focus of the work was shifted from Markovitz's theory onto *Risk-Parity allocation techniques*, where, as introduced in paragraph 2.1.3, the idea is to build the most diversified portfolio possible, allocating the budget on different assets in order to equally divide the portfolio risk. This in theory should produce portfolios where there are no riskier options and thus this allocation technique should be stabler where mean-variance would fail (for example with the failure of a low volatility stock).

Following this path, the best way to exploit the causal DAG structure in order to produce a risk-parity allocated portfolio was to work on *Graph clustering methods*. The idea is to cluster the causal graph to produce different sub-graphs with elements strongly connected among each other, and to allocate onto these clusters the budget B in a balanced way, to arrive to an optimal situation where diversification in the portfolio reaches its maximum. To do this, a variety of *spectral clustering* methods were considered, trying to exploit the properties of the *Graph Fourier Transformation*.

In these next paragraphs, we will present different methods to perform *spectral clustering* of both directed and undirected graphs, namely:

1. portfolio clustering through repeated portfolio cuts
2. direct evaluation of the *directed graph Laplacian* for portfolio cuts
3. study of a convex relaxation of the minimum cut problem in directed graphs

Finally, we will introduce the different allocation techniques we considered to equally divide the budget B among the different clusters obtained with methods (1), (2) and (3).

3.3.1 Graph Signal Processing and the study of Graph Fourier Transformations

Before beginning to explain the models proposed, it is needed to briefly present what exactly is spectral clustering and how *graph Fourier Transformation* (GFT) is built.

We can start by saying that spectral clustering it is obviously a branch of graph clustering methods, used to detect communities of nodes based on their mutual connections and edges. To do so, spectral clustering methods study signals f on the system in frequency domain, by exploiting both *eigenvectors* and *eigenvalues* of the *graph Laplacian matrix* L.

Each possible signal on the graph is defined as a mapping $f : V \to \mathbb{N}$, where V is the set of all vertices of a graph $\mathcal{G}(V, \mathcal{E})$. Basically f is a function that converts frequencies on each vertex of the graph into real numbers $f(i)$ and each signal f can be projected onto the eigenvectors of the graph Laplacian L, for this reason the study of the Laplacian eigenvectors via GFT is an important step in understanding how to cluster the graph $\mathcal{G}$.

We can now define the GFT as a transformation that eigendecomposes matrix L into its eigenvalues and eigenvectors, where eigenvalues represents frequencies on the graph, whether eigenvectors work as an orthonormal basis for the GFT, called the *Fourier basis*.

$$\mathcal{GF}[f](\lambda_l) = \hat{f}(\lambda_l) = < f, u_l > = \sum_{i=1}^{N} f(i) u_l(i) \tag{3.16}$$

where λ_l and u_l are respectively the $l - th$ eigenvector and eigenvalue of the graph Laplacian.

The study of frequency domain usually shows the clustering behavior of the network, but this formulation normally works for undirected graphs, when instead our goal is to present clustering methods for DAGs. Therefore, we need to find ways to calculate eigenvectors and eigenvalues for directed graphs in order to exploit spectral clustering principles even in this case.

3.3.2 Repeated Portfolio cuts for undirected graphs

The first and easiest way to divide a graph in a sequence of sub-graphs with strong internal connections is to exploit the *minimum cut clustering method* that tries to iteratively cut the graph into smaller partitions on the base of the *minimum cut algorithm*. We first consider a version of the algorithm suited only for undirected graphs, later to be used as another baseline for other clustering methods exploiting the directed nature of the causal graph.

Since we are in an undirected universe we can consider a graph $\mathcal{G} = \{V, \mathcal{E}\}$ with N vertices (representing factors and assets in our financial environment) and a *weight matrix* $\mathbf{W}$ can be defined, where each element of the matrix describes the connection between asset i and asset j

$$W_{ij} = \frac{|\sigma_{ij}|}{\sqrt{\sigma_{ii}\sigma_{jj}}} = |\rho_{ij}| \tag{3.17}$$

where σ_{ij} is the correlation between assets i and j, whether σ_{ii} and σ_{jj} are the respective variances of assets i and j [9]. Basically, we are defining the weight matrix of an undirected graph structure based solely on the correlations between sampled data for the different factors and assets. The resulting matrix $\mathbf{W}$ is symmetric, this means that it perfectly represent an undirected graph (where all connections are symmetric) and that $\mathbf{W} = \mathbf{W}^T$.

Having defined this undirected graph structure, we can now try to cluster it trying to minimize what is defined as the *cut size function* of the graph [9]:

$$Cut(V_1, V_2) = \sum_{i \in V_1} \sum_{j \in V_2} W_{ij}. \tag{3.18}$$

This function defines a cut on the graph, i.e. the removal of a certain amount of edges that divides graph $\mathcal{G}$ into two distinct sub-graphs V_1 and V_2; each cut is the sum of all the weighted edges that it eliminates (W_{ij}, meaning that its minimization would also minimize the number and the importance of the deleted edges, bringing us to the two best possible partitions for graph $\mathcal{G}$.

However, this minimization process could frequently lead to a situation of isolated nodes, thing that could impact negatively the budget allocation to be performed later on, and for this reason we opted to utilize a normalized version of the cut size function that adds a term in order to guarantee that the two sub-graphs

V_1 and V_2 are as large as possible:

$$CutN(V_1, V_2) = \left(\frac{1}{N_1} + \frac{1}{N_2}\right) \sum_{i \in V_1} \sum_{j \in V_2} W_{ij} \tag{3.19}$$

where N_1 and N_2 are the number of nodes in sub-graphs V_1 and V_2 respectively; in this way the minimization problem adds a term that reaches its minimum for $N_1 = N_2 = \frac{N}{2}$ meaning that we are enforcing the two subsets to be as similar as possible in term of dimension [9].

Minimum cut and minimum normalized cut are NP-hard problems, meaning that their exact numerical solution becomes unfeasible with standard methods when N grows too much, since the number of possible cuts in a graph with N nodes is equal to $2^{(N-1)} - 1$ that becomes incredibly large just with $N > 100$, situation that usually occurs in real-world scenarios.

Though, we know that there exists an approximate solution for the minimum normalize cut in the frequency domain [9], given by

$$CutN(V_1, V_2) = \frac{\mathbf{x}^T \mathbf{L} \mathbf{x}}{\mathbf{x}^T \mathbf{x}} \tag{3.20}$$

where L is the graph's Lagrangian and $\mathbf{x}$ is an indicator vector defined as

$$x(n) = \begin{cases} \frac{1}{N_1}, & for n \in V_1, \\ -\frac{1}{N_2}, & for n \in V_2, \end{cases} \quad . \tag{3.21}$$

Given this, the minimization of equation (3.20) can also be written as

$$\min_x \mathbf{x}^T \mathbf{L} \mathbf{x} \quad s.t. \quad \mathbf{x}^T \mathbf{x} = 1 \tag{3.22}$$

and can be solved with the eigendecomposition $\mathbf{L}\mathbf{x} = \lambda_k \mathbf{x}$ of Lagrangian $\mathbf{L}$ into its eigenvectors $\mathbf{X}$ and its eigenvalues λ_k.

Since we are in an undirected situation, finding the graph Laplacian is pretty straightforward, since it can be derived from the symmetric weight matrix $\mathbf{W}$ and the so called *Degree matrix* $D \in \mathbb{R}^{N \times N}$, diagonal with elements defined as

$$D_{ii} = \sum_{n=1}^{N} W_{in} \tag{3.23}$$

where element i of the diagonal is the sum of all the connections of node i with all other nodes in the network.

From this, we can define the Laplacian matrix for an undirected graph as

$$L = D - W. \tag{3.24}$$

Once the graph Laplacian is defined, it is possible to proceed with its eigendecomposition, finding its eigenvectors $u_1, \ldots, u_l$, that we consider to find the best possible solution for equation (3.20) through (3.22), where we can see that optimal values for indicator vector $\mathbf{x}$ are exactly the eigenvectors $\mathbf{u}$ of the graph Laplacian.

Skipping the obvious basic solution in the form of the first constant eigenvector u_1 (that has all elements equal, not being able to define any partition), we can find $\mathbf{X}_{opt}$ in the form of the second eigenvector u_2, also called *Fielder eigenvector* [9].

Knowing that indicator vector $\mathbf{x}$ has the form expressed in equation (3.21), it is possible to define the two optimal partition on the base of the *sign* function of $x_{opt} = u_2$ obtaining a straightforward way to allocate each vertex in its best subgraph:

$$sign(x(n)) = \begin{cases} 1, & \text{for } n \in V_1, \\ -1, & \text{for } n \in V_2, \end{cases} \quad . \tag{3.25}$$

Once obtained the two partitions V_1 and V_2 we simply choose the bigger one and we restart the whole process, continuing to cut the biggest sub-graph in the set until the desired number of cuts c is reached and $c - 1$ sub-graphs are obtained. The whole process can be reported in algorithms (3.1) and (3.3).

3.3.3 Repeated portfolio cuts for directed graphs

What we did until now was to find balanced partition in a graph based on just correlation measures, used to define the weight matrix W for the fully connected undirected graph used to describe the data.

This approach has the same problems encountered when dealing with standard mean-variance optimization, i.e. the method only considers a graph made of assets, without the presence of financial factors, the fact that we rely solely on correlations measures to find the different partitions and, following this, the fact that we obtain a fully connected undirected graph, encoding even the smallest covariance values into edges that don't reflect real connections between assets.

To deal with these critical issues the best thing to do would be to find a way to work on sparser matrices, with a better encoding of the real causal connections

Algorithm 3.1 Single portfolio cut

input: weight matrix W
output: V_1 and V_2 subsets of input graph
Obtain D from W
$D, W \rightarrow L = D - W$
eigendecomposition of L
$x_{opt} \leftarrow u_2$
for node i in *original graph*
  if $x_{opt}[i] > 0$
    Node $i \in V_1$
  else
    Node $i \in V_2$
  end if
end for

Algorithm 3.2 Repeated portfolio cuts

input: W (weight matrix) and c (number of cuts)
output: V_1, V_2, ..., V_{c+1} sub-graphs
create empty *list*
list gets $\mathcal{G}$
for i in c
  Check for the biggest sub-graph in *list*
  Find $W_{biggest}$
  $V_1, V_2 \leftarrow$ SinglePortfolioCut($W_{biggest}$)
  list **add** V_1, V_2
  list **remove** $\mathcal{G}$
end for

between assets and factors, but the method presented in the previous paragraph 3.3.2 is designed to work on symmetric matrices (such as undirected version W) in order to find the graph Laplacian L with the formula $L = D - W$, whereas weight matrix C of the causal graph is asymmetric due to direct connections being asymmetric in nature.

For this reason, we decided to work on ways to symmetrize directed weight matrix C, in order to exploit algorithms (3.1) and (3.2) for directed graph clustering tasks.

The first type of symmetrization considered is of the type $\mathbf{C} + \mathbf{C}^T$ [10][11], that produces a symmetric weight matrix C_{sym} for a new undirected graph. In this case the number of edges is maintained, and the directionality information contained in the edges of the original DAG is encoded in the new weights of the graph obtained.

However, this type of matrix symmetrization could not account well for another important features normally used to cluster directed graphs, i.e. the number and the identity of in and out edges of a node. In fact, to cluster directed networks, we should consider not just the direct connections between two nodes i and j, but also if these nodes have one or more common parents or children.

For this reason we moved onto a different type of matrix symmetrization that would also account for the in and out edges of each node of the network. Authors Satuluri and Parthasarathy [11] considered two different matrices:

1. $\mathbf{A} = \mathbf{C}\mathbf{C}^T$ called *bibliographic coupling matrix*, trying to capture similar outgoing edges from each pair of nodes
2. $\mathbf{B} = \mathbf{C}^T\mathbf{C}$ called *co-citation strength matrix*, capturing similar incoming edges for each pair of nodes.

Combining these two matrices we obtain

$$\mathbf{C}_{bib} = \mathbf{A} + \mathbf{B} = \mathbf{C}\mathbf{C}^T + \mathbf{C}^T\mathbf{C} \tag{3.26}$$

called *Bibliometric Symmetrization*. In this way, with the application of previous algorithms we should be able to obtain clusters based on the causal structure of the graph instead on just pure correlation measures as before.

3.3.4 Estimating the graph directed Laplacian

In the previous clustering methods, we worked on a symmetrized undirected representation of the DAG encoding the cause-effect relations of the system under analysis. The two symmetrization methods try to encode causal information into

undirected graphs, to exploit the fact that in symmetric networks it is extremely easy to derive the graph Laplacian (and as a consequence its eigenvalues and eigenvectors with whom to perform the clustering algorithms).

This time, instead of working on weight matrix C, we will propose a formulation to derive what it's called the *Directed Graph Laplacian* L_d [10][12].

The formulation for a directed version of the graph Laplacian was initially proposed by Chung [12] where he defined L_d as

$$\mathbf{L}_d = \mathbf{I} - \frac{\mathbf{\Pi}^{\frac{1}{2}}\mathbf{P}\mathbf{\Pi}^{-\frac{1}{2}} + \mathbf{\Pi}^{-\frac{1}{2}}\mathbf{P}^T\mathbf{\Pi}^{\frac{1}{2}}}{2}. \tag{3.27}$$

Matrix $\mathbf{P}$ is the so called *transition matrix* and is defined on the base of the weight matrix C and the outgoing connections of each node in the graph. Its elements can be calculated as

$$P_{ij} = \frac{C_{ij}}{K_i^{out}}. \tag{3.28}$$

Matrix $\mathbf{\Pi}$ instead, is the diagonal matrix representing the stationary distribution of a random walk performed on the DAG under analysis

$$\mathbf{\Pi} = \texttt{diag}(\pi_1, ..., \pi_n) \tag{3.29}$$

where π_i element of the diagonal is the probability to find the random walk on node i after a time t, with $t \to \infty$.

The important point about this formulation is that it respects the so called *Cheeger inequality* [12]

$$2h(\mathcal{G}) \geq \lambda \geq \frac{h^2(\mathcal{G})}{2\Delta(\mathcal{G})} \tag{3.30}$$

where $\Delta(\mathcal{G})$ is the maximum number of connections that any node can have in graph $\mathcal{G}$, λ is the *spectral gap* of the Lagrangian (the distance between its two biggest eigenvalues) and $h(\mathcal{G})$ is the so called *Cheeger constant* defined as

$$h(\mathcal{G}) := \min\left\{\frac{|\partial A|}{|A|} \text{ s.t. } A \in V, 0 < |A| \leq \frac{1}{2}|V|\right\} \tag{3.31}$$

with V the set of all vectors in $\mathcal{G}$, A as a subset of V and $|\partial A|$ as a set of edges in $\mathcal{E}$ going from a node in A to a node outside A (in $V \backslash A$).

Basically, Cheeger constant can be seen as a measure of the clustering behavior of the network under analysis and the fact that formulation (3.27) respects inequality (3.30) means that it is suited for a clustering analysis of the graph.

Knowing this, we can integrate the calculation of L_d in algorithm (3.2) in order to iteratively find a good set of partitions for graph $\mathcal{G}$.

3.4 Graph Fourier Basis through Lovász extension of cut size set function

Until now, the different methods considered for graph clustering worked on trying to find the graph Laplacian (directed or undirected) in order to exploit a convex relaxation of the graph cut size function (3.20) to find the best division of vertices into the different clusters V_1 and V_2. From here, these methods used an iterative approach, by searching at every iteration for the biggest sub-graph in the set, dividing it into two smaller partitions and continuing the process until the desired number of cuts was reached.

This approach has obviously its advantages, since it is easy to solve, relatively fast and it's easily implemented in any context, but working on a different sub-graph each time could be a way to miss on important information about cause-effect relations between elements in different sub-graphs, even if causal information should be stored in the directed version of the graph Laplacian L_d or in the symmetrized version of the weight matrices.

In this section we want to propose a method for graph clustering that aims to extract a Graph Fourier Basis from a convex extension of the cut size function (3.18), in order to use it to perform a *k-way* clustering of the graph $\mathcal{G}$, separating it in k partitions without cutting the graph in an iterative manner.

3.4.1 Sub-modular set functions and cut size Lovász extension

Let us recall the cut size function that was introduced in section 3.3.2:

$$\text{cut}(\mathcal{S}, \bar{\mathcal{S}}) := \sum_{i \in \mathcal{S}, j \in \bar{\mathcal{S}}} w_{ij} \tag{3.32}$$

where w_{ij} is the edge connecting vector i with vector j.

We know that equation (3.32) is a *set function*, i.e. a function that has as domain a family of subsets of a given super-set (in this case $2^{\mathcal{V}}$, the set of all the possible subsets of vertices in $\mathcal{G}$). Also, it is known that cut size function is a so called *sub-modular set function*, whose definition is given below [13]

> A set function $F : 2^{\mathcal{V}} \rightarrow \mathbb{R}$ is sub-modular if and only if, $\forall \mathcal{A}, \mathcal{B} \subseteq \mathcal{V}$, it satisfies the following inequality:
>
> $$F(\mathcal{A}) + F(\mathcal{B}) \geq F(\mathcal{A} \cup \mathcal{B}) + F(\mathcal{A} \cap \mathcal{B}). \tag{3.33}$$

With this knowledge, we can introduce the so called *Lováz extension* of a generic set function [13]:

> Let $F : 2^{\mathcal{V}} \rightarrow \mathbb{R}$ be a set function with $F(\emptyset) = 0$. Let $x \in \mathbb{R}^N$ be ordered in increasing order such that $x_1 \leq x_2 \leq \cdots \leq x_N$. Define $C_0 \triangleq \mathcal{V}$ and $C_i \triangleq \{j \in \mathcal{V} : x_j > x_i$ for $i > 0$. Then, the Lovász extension $f : \mathbb{R}^N \rightarrow \mathbb{R}$ of F, evaluated at x, is given by:
>
> $$\begin{aligned} f(x) &= \sum_{i=1}^{N} x_i (F(C_{i-1}) - F(C_i)) \\ &= \sum_{i=1}^{N-1} F(C_i)(x_{i+1} - x_i) + x_1 F(\mathcal{V}). \end{aligned} \tag{3.34}$$

Now, a very important property of every sub-modular function is the fact that its Lovász extension is a convex function (thus very easy to optimize) and we know that the optimization of a set function $F(\mathcal{S})$ in its domain it's equivalent to the optimization of its Lovász extension on an hypercube $[0, 1]^N$ [13]. Formally

$$\min_{\mathcal{S} \subseteq \mathcal{V}} F(\mathcal{S}) = \min_{X \in [0,1]^N} f(x). \tag{3.35}$$

Basically, to find the set $\int$ that minimizes set function $F(\int)$ it is equal to minimize $f(x)$ on an hypercube with dimension N.

In [13], the Lovasz extension of cut size function for a directed graph was defined as the *Graph directed variation* (GDV) of graph signals

$$f(x) = \sum_{i,j=1}^{N} w_{ji} [x_i - x_j]_+ := GDV(x) \tag{3.36}$$

where $[y]_+ := max\{0, y\}$.

Given this definition of the Lovász extension of the cut size function, we can minimize the graph directed variation defined before in order to obtain the set of N orthonormal vectors of the Graph Fourier basis $\mathbf{X} := (x_1, x_2, ..., x_N) \in \mathbb{R}^{N \times N}$ of the system with this minimization problem:

$$\begin{aligned} \min_{\mathbf{X} \in \mathbb{R}^{N \times N}} GDV(\mathbf{X}) &:= \sum_{k=1}^{N} GDV(x_k) \\ \text{s.t. } \mathbf{X}^T\mathbf{X} = \mathbf{I}, \quad & x_1 = b\mathbf{1} \end{aligned} \tag{3.37}$$

where the two constraints are posed in order to assure that the basis is orthonormal and to avoid the trivial null solution. This, even if the Lovász extension is defined convex for a sub-modular function, it is still a non-convex problem due to the constraints we posed to grant orthogonality.

Thus, a proper method is required to solve this problem and to obtain the graph Fourier basis needed for the clustering task. Among all the possible choices, we decided to adopt the *Splitting Orthogonality Constraints method*, also called *SOC method*, an algorithm based on the alternating method of multipliers (ADMM) that aims to solve the problem of non-convexity of the constraints by iteratively splitting them with the aid of an auxiliary variable $\mathbf{P}$ [13].

The convergence of this method has still to be mathematically demonstrated, but it has shown good results in practical situations and it deals directly with the main problem of equation (3.37), i.e. the non-convex orthogonality constraints.

3.4.2 K-WAY CLUSTERING ALGORITHM

As we introduced at the beginning of section 3.4, the idea is to produce a *k-way* clustering algorithm, in order not to work on separated sub-graphs, but to produce k partitions directly from the original graph $\mathcal{G}$.

Now, we know that matrix $\mathbf{X}$ of orthonormal vectors obtained with equation (3.37) can be considered as the graph Fourier basis of the network. Columns of $\mathbf{X}$ can be then viewed as eigenvectors of the graph Laplacian, with their associated eigenvalues, representing frequencies in the spectral domain we are working on.

We can then consider each row i of matrix $\mathbf{X}$ as a vertex in the graph $\mathcal{G}$ and each element of this row can be viewed as a feature associating the i-th vertex with each one of the eigenvalues related to the N eigenvectors of the graph Laplacian; remembering that each eigenvalue $\lambda_1, ..., \lambda_N$ represent a frequency in the frequency domain, we can account for the first k features for each vertex in order to obtain k clusters in the network, exploiting the *k-means clustering algorithm* [14].

Thus, to produce k clusters in the graph we just need to look at the first k eigenvalues of the graph Laplacian, in this case obtained with the minimization of the Lovász extension of the cut size function.

The entire procedure is reported in algorithm (3.3).

Algorithm 3.3 k-way directed graph clustering

use SOC optimization to derive $\mathbf{X}$ matrix of eigenvectors
find the first k eigenvectors $u_1, \ldots, u_k$
define features for the different vertices $V_1, \ldots, V_K$ based on $u_1, \ldots, u_k$
use K-MEANS clustering to obtain k subsets

3.5 Budget allocation schemes

Finally, it is possible to talk about budget allocation schemes for spectral clustering methods. Till now, the only result obtained by the different clustering methods was to separate the original causal graph into small clusters supposedly based on causal relations between contained assets and factors, but to obtain a proper allocation vector w, it is needed a defined rule to divide the budget B among the k clusters obtained. To do this, we decided to tackle the problem in two different ways [9]:

1. **Allocation 1** $W_i = \frac{1}{2^{K_i}}$, where K_i is the number of cuts needed to obtain subgraph $\mathcal{G}_i$. Basically at each iteration of the clustering algorithm we keep track of the number of cuts used to reach that specific situation and later we use that information to allocate the right amount of budget onto i-th asset; this means that clusters that are very different from the rest of the graph are splitted earlier and will receive a greater percentage of budget respect to the ones that are close together. This method is the more articulate of the two and it tries to create a particular hierarchy among the sub-graphs in order to correctly allocate the budget B.
2. **Allocation 2** $W_i = \frac{1}{K+1}$, where K is the total number of cuts performed to obtain $K+1$ sub-graphs $\mathcal{G}_i$. This is a simpler method, more suited for cases where there is not a true hierarchy among sub-graphs, such as in the k-way clustering based on the Lovász extension of the cut size, where all the clusters are obtained at the same time using a K-means clustering algorithm.

Once obtained the designed weight W_i for each subgraph, we just need to divide this percentage of budget equally among all the members of the cluster, finally obtaining an allocation vector $\mathbf{w} = w_1, w_2, \ldots\ldots, w_{N_a+N_f}$.

Since the graph consists of both investible assets and non investible factors, there is the necessity to re-allocate the percentage of budget of factors F onto different assets A; to do this we decided to split the budget of a specific factor among all the assets present in its cluster. In the rare case of a cluster presenting only factors it is needed to split the budget between all the assets with whom a specific factor has connection with, in order to respect the necessity to have a fully invested portfolio ($\sum_{i=1}^{N_a} w_i = 1$).

4

Model results and statistical analysis

Up to this point, we worked on a theoretical definition of the different models, trying to exploit information derived from the causal structure underlying the available data, now the focus will be on the analysis of the models presented in chapter 3, conducting out-of-sample tests on randomly generated datasets in order to see their performance on new and unseen data. Also, we will try to test the same models on new datasets obtained from different intervention distributions, constructed on the basis of the original SCM $\mathfrak{C}$, performing one or more hard/soft interventions on the system under analysis.

Behind this idea there is the will to test if the models proposed can have some advantages over the benchmark ones in situations where a sudden modification of the market occurs, modification modeled with the aid of interventions on the static system.

This chapter will be divided into three sections: In the first one, the dataset used for calculating asset allocation vectors w will be introduced together with the different metrics considered for the analysis of the performance on the sample data of the models, while in the second part we will effectively test the models to assess their performance on the static system they were trained on. Finally, in the third section, intervention distributions will be introduced and the models will be tested on newly sampled datasets based on different types of interventions performed on the original SCM, in order to see if any advantage over the benchmark can be found when in presence of unexpected and sudden changes.

4.1 Datasets and metrics used for model testing

In paragraph 3.1.3 we introduced the methodologies used for randomly sampling the data points representing asset returns, used in the presented models to obtain asset allocation vectors w. These data were sampled on the basis of an weight matrix of the causal DAG derived from the SCM, graph that in this specific case was composed of $N = 27$ nodes, divided into $N_a = 18$ investible assets A and $N_f = 9$ non-investible factors F.

The size of the graph and the exact composition of assets and factors was determined randomly, with the only constraint of having the number of assets at least double the number of factors, in order to avoid as much as possible the situation of finding clusters in graph $\mathcal{G}$ with only or mostly factors, causing the problem of having to divide the allocated budget into different subsets, mining the ability of the model to correctly diversify the investments. Also, such a number of nodes was chosen to ease the computations and speed up the process of model-testing, while still giving access to graph-clustering possibilities.

After determining the values for N_a and N_f, the total length of the time series to sample was defined. We decided to sample it in terms of number of trading years, so a number of trading days equal to a multiple of 252 was sampled. A total of 2520 points were sampled from the distribution, later to be divided into two parts, with $252 \times 8 = 2016$ points for the training set and the remaining 252×2 points for the testing set.

Also, as we already introduced in paragraph 3.1.3, since the idea is to directly sample the daily returns for each asset, it is possible to assume each noise factor $Z_i, ..., Z_{N_a+N_f}$ to be distributed as a random normal $\mathcal{N} \sim (\mu, \sigma^2)$ with $\mu = 0$ and $\sigma^2 = 1$.

Given this it is possible to extract from matrices C and $\mathbf{Z}$ a number T of samples representing a simulation of real world asset returns, with a time-wise distribution visible in figure 4.1 and 4.2, representing the instantaneous daily returns and the cumulative returns of random assets of the system under analysis.

After the definition of the dataset it is now possible to introduce the different metrics later to be used to estimate the results of the various models.

4.1.1 Metrics

After defining on what data the models will be tested on, we need to define what are the metrics that will be considered in order to evaluate the performance of the

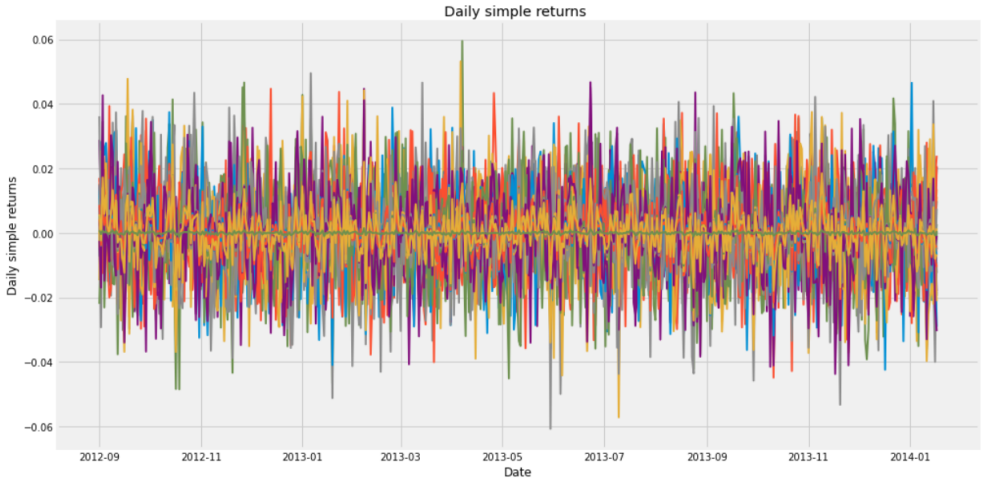


Figure 4.1: graphical representation of the sampled daily returns for assets $A_1, \ldots, A_18$ and factors $F_1, \ldots, F_9$

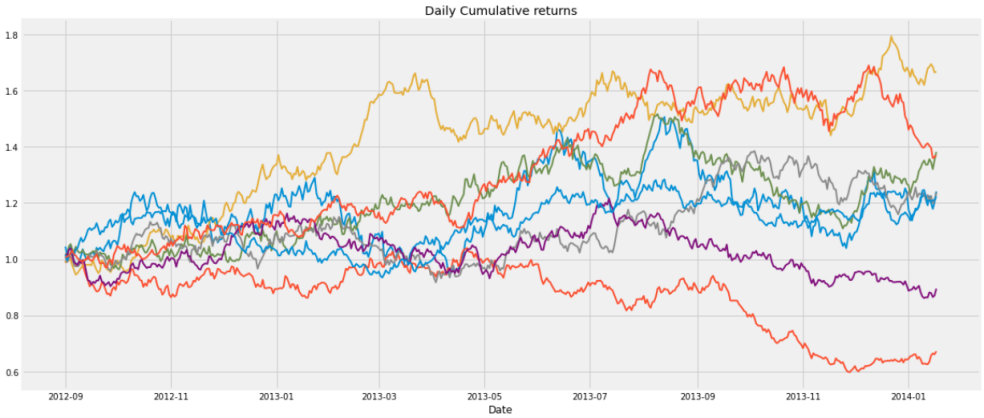


Figure 4.2: graphical representation of the cumulative returns for some of the assets of the network (in this case $A_1, \ldots, A_9$)

obtained portfolios, with the aim to detect any difference that could confirm the better performance of a model over the others.

To do this, three metrics were considered:

1. **Sharpe Ratio** This metric was already introduced earlier in chapter 3 with equation (3.11), but it will be recalled here with a slightly different form:

$$\text{Sharpe Ratio} = \frac{\mathbb{E}[R_X] - R_f}{\text{StdDev}(R_X)} \tag{4.1}$$

where $\mathbb{E}[R_X]$ is the expected return of the portfolio under analysis, R_f is called the *risk free rate* and $\text{StdDev}(R_X)$ is the standard deviation of the port-

folio returns. This metric is pretty straightforward, since it is directly related to the maximization of the expected return with at the same time the reduction of the risk associated to the portfolio (represented by the standard deviation). Basically, Sharpe ratio defines when an investment has a positive risk/reward ratio, and values of this metric over 1 are considered desirable in financial analysis, since it indicates that a particular investment correctly balances its volatility with a good level of expected return.

Alone, this metric is not enough to define the goodness of a specific portfolio, since it just returns an indication of the relation between these two values, but this only explains how much excess return the investor gets for any unit of risk considered, not how risky or how rewarding a specific set of investments is overall. In practice, two very different portfolio could have the same exact Sharpe ratio, one with high volatility and high expected return and one with both values really low, and the only Sharpe analysis wouldn't be able to provide enough information to choose between the two models.

2. **Volatility** This is simply the standard deviation of the portfolio under analysis. Normally, the daily standard deviation is considered and then is multiplied by $\sqrt{T}$, with T as the length of the testing period considered, in order to scale the volatility to the entire time window. This metric can give us an idea on how risky is a specific asset or on what type of portfolio we are investing on (a risky one or a safer one). Volatility and Sharpe ratio combined can return an interesting overview on the constructed portfolio in terms of risk and reward obtained.

3. **Maximum Drawdown** MDD is a measure of the riskiness of the portfolio allocation and it is calculated considering the maximum observed loss from a peak to a trough of a portfolio in a time window T, it represents the downside risk of the portfolio in a specific time window:

$$MDD = \frac{\text{Trough value} - \text{Peak Value}}{\text{Peak value}}. \tag{4.2}$$

With the analysis of the volatility of a portfolio it is possible to see how risky it is a certain investment, but standard deviation can be negative or positive, and a risky asset can also provide the investor with big returns, whether the measure of maximum drawdown only indicate what is the biggest loss that can occur. It is calculated considering the peaks and the troughs of the cumulative returns of the portfolio under analysis.

This will be the metrics later to be used to evaluate the performances of the different asset allocation techniques, both in presence or in absence of interventions on the original dataset.

4.2 Model testing on unseen data

In this paragraph we will test the models presented in chapter 3 on a small subset of testing data obtained during the sampling phase, equal to two trading years and with $T = 252 \times 2$ points. This should display how the different portfolios works in a static environment, where the causal ordering and the parental relations between assets and factors are considered as constant and where each asset or factor has approximately the same normal distribution.

In order to obtain significant results, a total of $n = 1000$ repeated trials will be considered, in which all the different models will be trained and tested against the same train and test sets. More precisely, at each iteration a new dataset with 252×10 elements is created and divided into training and test set; each portfolio is then optimized on the new training data and tested against the new test set.

Finally each one of the previously introduced metrics is calculated for each different portfolio and it is stored inside a new dataframe. At the end of the n iterations, the statistical distribution of the different metrics is considered in order to make confrontations between all the models under analysis.

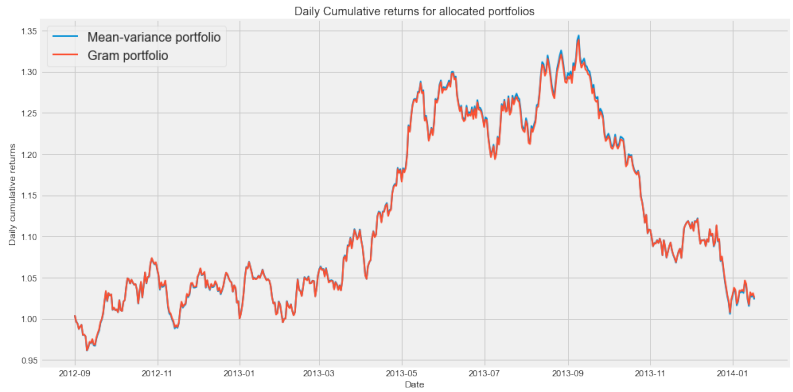
It is necessary to state that since all of the models will be tested against data sampled on the base of very general assumption (similar distribution of returns for example), our main interest is to consider the results in relative terms, without looking too much at the absolute values obtained for Sharpe ratio, volatility and maximum drawdown, that could change drastically when tested in a real world scenario.

4.2.1 Sharpe ratio

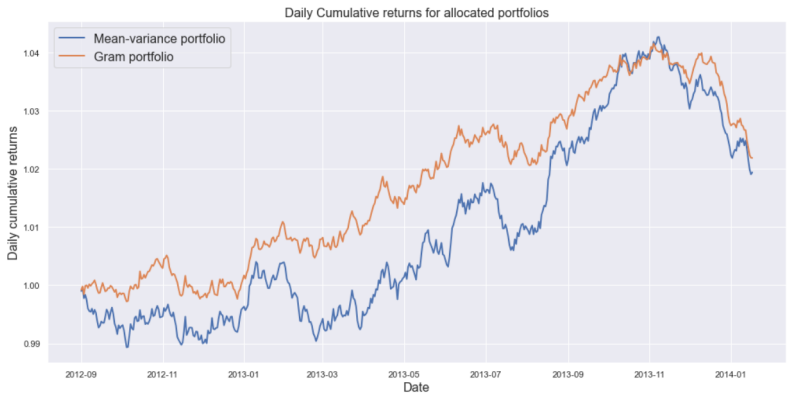
The first metric we will consider in order to evaluate the different models after $n = 1000$ trials is the Sharpe ratio; this will give an indication about what portfolio is more suited to balance the maximization of the expected return with the minimization of the portfolio volatility.

But first it is interesting to look at some single trial results. In figures 4.3a and 4.3b are represented some single trials obtained with Mean-variance Markovitz's portfolio and of its realization exploiting the Gram matrix G.

These are the cumulative returns obtained multiplying the allocation vector of the two portfolios with the daily returns of the sampled assets A in order to obtain the daily return for each one of the two models; the two single trials were obtained by training and testing the models against the dataset introduced previously in paragraph 4.1, with 27 total nodes (fig. 4.3a) and against a new sampled dataset



(a) 27 total nodes



(b) 450 total nodes

Figure 4.3: Daily cumulative returns of Markovitz's mean-variance model and Gram matrix model over two different networks with low (a) and high (b) number of nodes

with 450 total nodes (fig. 4.3b). It is easy to see that, as predicted, the two portfolios obtain the same cumulative returns when the system is composed of a low number of assets, while with greater dimensions (such as 450) the two methods have clear discrepancies given by the increasing difficulty to correctly calculate the covariance matrix Σ.

Given this, in figure 4.4 we present the Sharpe ratio trend for the mean-variance and the Gram matrix models, calculated over the 1000 trials described above.

In this case, since the network has only a total of 27 nodes (fig. 4.3a), as expected the two profiles obtained are almost identical.

We can look instead at the results obtained considering the repeated cuts methods, employed both in undirected and directed forms and with both the symmetrizations considered (standard and bibliometric) for the direct case; for these three portfolios both methods for budget allocation described in section 3.5 were ex-

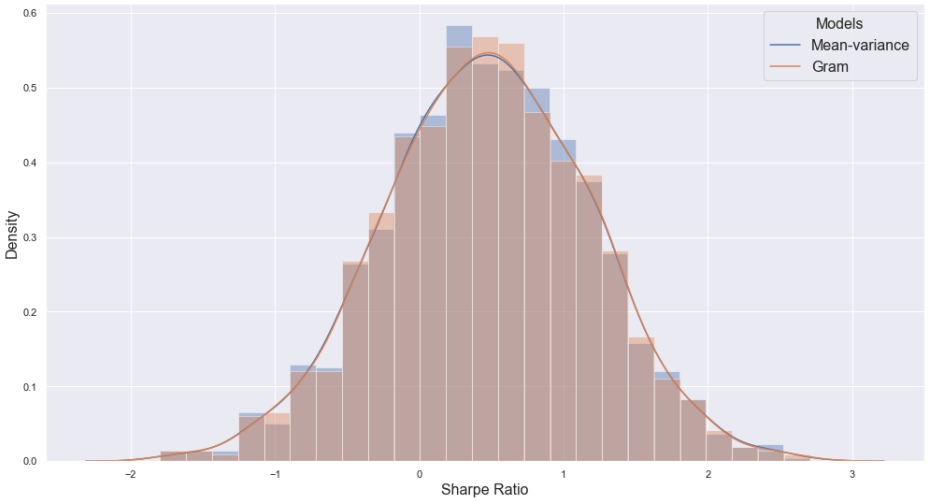


Figure 4.4: Sharpe ratio trend over 1000 repeated trials for mean-variance models

	Mean-Variance	Gram
Mean	0.470913	0.0.471578
σ^2	0.502969	0.501897

Table 4.1: Sharpe ratio statistics for Mean-variance and Gram models

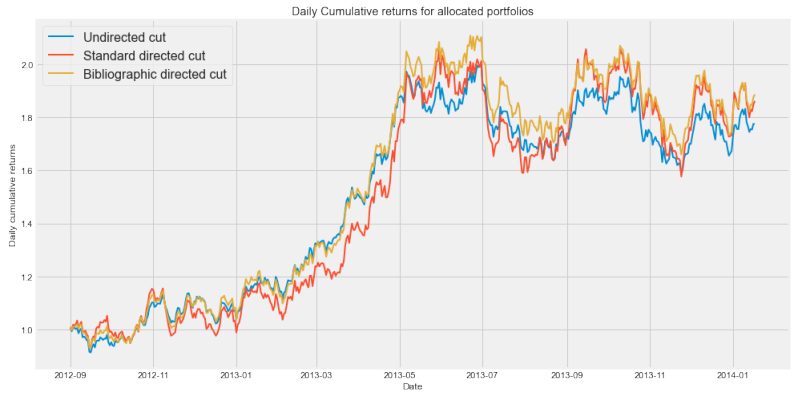
ploited, calling **Allocation 1** the pseudo hierarchical method $W_i = \frac{1}{2^{K_1}}$ (fig. 4.5a) and with **Allocation 2** the equal allocation method $W_i = \frac{1}{K+1}$ (fig. 4.5b).

From figure 4.6 we can take a look at the Sharpe ratio trend for the three different methods with both allocation schemes considered; Also looking at the results reported in table 4.2, it is easy to see how practically all the versions of the cut size method obtain slightly better results than the standard mean-variance in terms of mean value of Sharpe ratio after repeated trials, with practically the same value of variance.

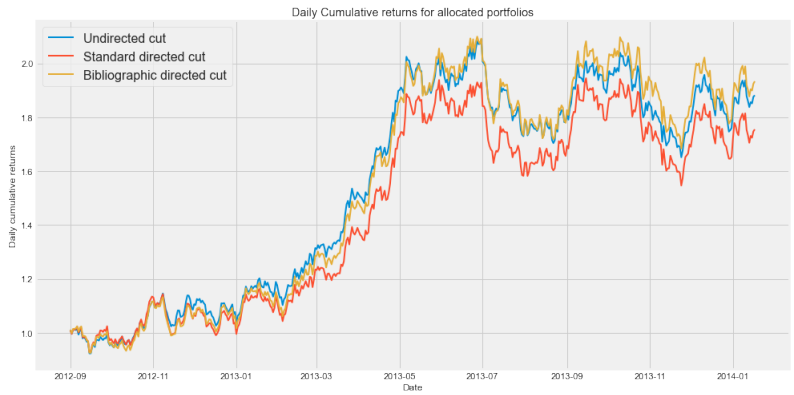
	Cut1	Cut2	D.Cut 1	D.Cut 2	B.Cut 1	B.Cut 2
Mean	0.493988	0.496226	0.491990	0.490848	0.490855	0.488683
σ^2	0.499332	0.499188	0.499803	0.498675	0.499727	0.498157

Table 4.2: Sharpe ratio statistics for 2-way clustering methods

Finally we can take a look at the results in terms of cumulative returns and Sharpe ratio for the Lovász extension cut method and the directed Laplacian cut method in figures 4.7a and 4.7b. Also in this case no clear differences can be noted in the static case, where k-way Lovász extension algorithms seems to have a slight edge in terms of Sharpe ratio performances over the other 2-way clustering methods,



(a) Allocation 1



(b) Allocation 2

Figure 4.5: Daily cumulative returns of the Undirected, standard directed and bibliometric directed cut methods with both allocation methods

but in such a static case all the differences are minimal and still the overall shape is almost the same.

	Directed Laplacian cut	Lovász extension cut
Mean	0.499410	0.483163
σ^2	0.500032	0.493228

Table 4.3: Sharpe ratio statistics for directed Laplacian and Lovász extension methods

We can then summarize all the results in terms of Sharpe ratio in figures 4.8 and 4.9, where it is possible to see the different trends, the mean values and the variances of the Sharpe ratio for different portfolios allocations.

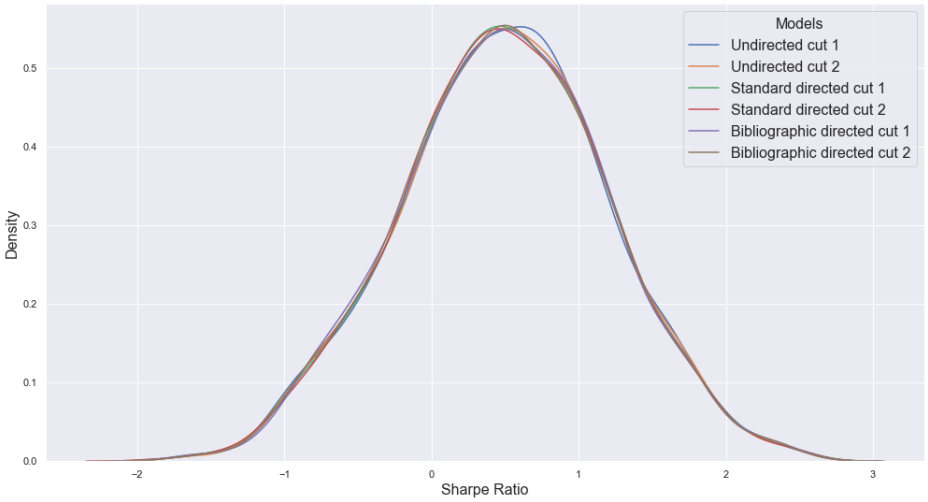


Figure 4.6: Sharpe ratio trend over 1000 repeated trials for 2-way clustering algorithms

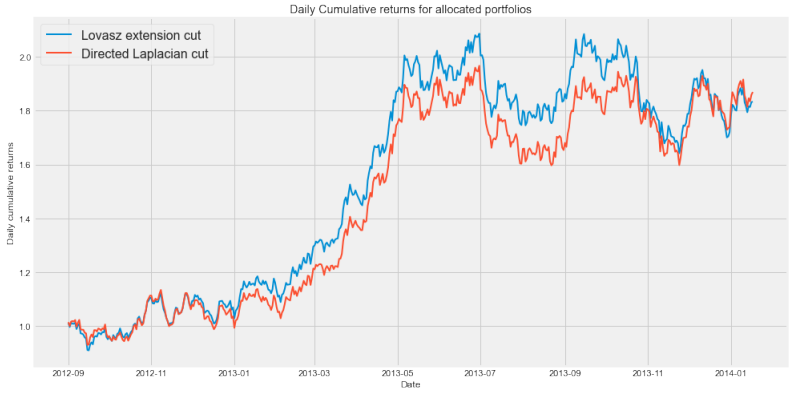
4.2.2 Volatility

Just analyzing the Sharpe ratio is not enough to assess the robustness of the different portfolios and for this reason it is needed to consider other two metrics introduced in section 4.1.1, namely Volatility and Maximum drawdown.

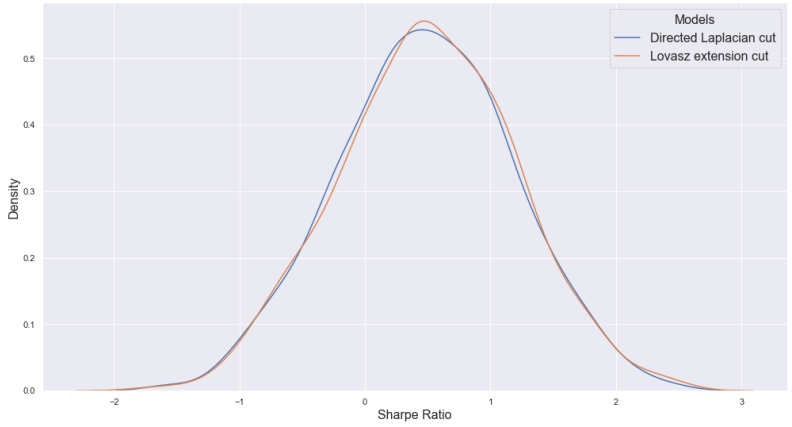
As you can see in figure 4.10, while in terms of Sharpe ratio the values were very similar for all the different models, in this case it is possible to notice immediately some clear differences between the portfolios under analysis.

It is clearly visible the different shapes of the volatility trends for three macrogroups of models:

- Mean-variance models, as you can see from tables 4.4 and 4.5 and figure 4.11, have a lower volatility mean value respect to other models, but at the same time have a flatter distribution, providing the highest encountered value in terms of variance.
- Models based on undirected cuts and on the Lovász extension have higher values in terms of volatility mean values, but their distribution is more regular and with a lower variance.
- Models based on directed cuts (with both standard and bibliometric symmetrizations) and directed Laplacian instead, while obtaining similar values to other clustering methods in terms of volatility mean values, have the best results in term of volatility variance, with very peaked distributions (as you can see in fig. 4.10)



(a) Daily cumulative returns



(b) Sharpe ratio trend

Figure 4.7: Daily cumulative returns plot (a) and Sharpe ratio trend (b) for Directed Laplacian and Lovász extension cuts models

	Mean-Variance	Gram	Directed Laplacian	Lovász
Mean	28.02	27.97	37.09	35.90
σ^2	59.72	58.87	1.37	8.04

Table 4.4: Volatility statistics (1)

	Cut1	Cut2	D.Cut 1	D.Cut 2	B.Cut 1	B.Cut 2
Mean	36.20	36.30	36.27	37.24	37.95	35.81
σ^2	7.49	5.73	1.30	1.37	1.40	1.26

Table 4.5: Volatility statistics (2)

4.2.3 Maximum drawdown

Next, it's possible to analyze in the same way the maximum drawdown metric, in order to better understand what is the maximum drop in returns that it is ex-

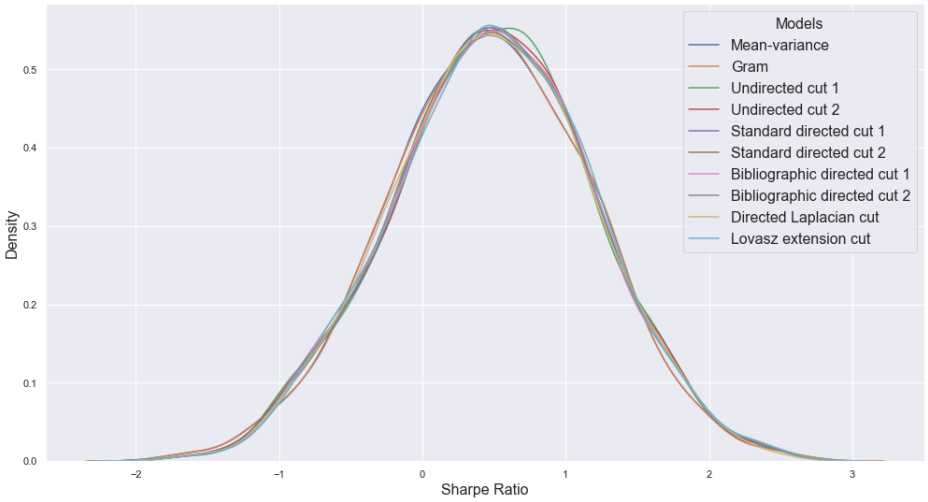


Figure 4.8: Sharpe ratio trend over 1000 repeated trials for all the 10 models

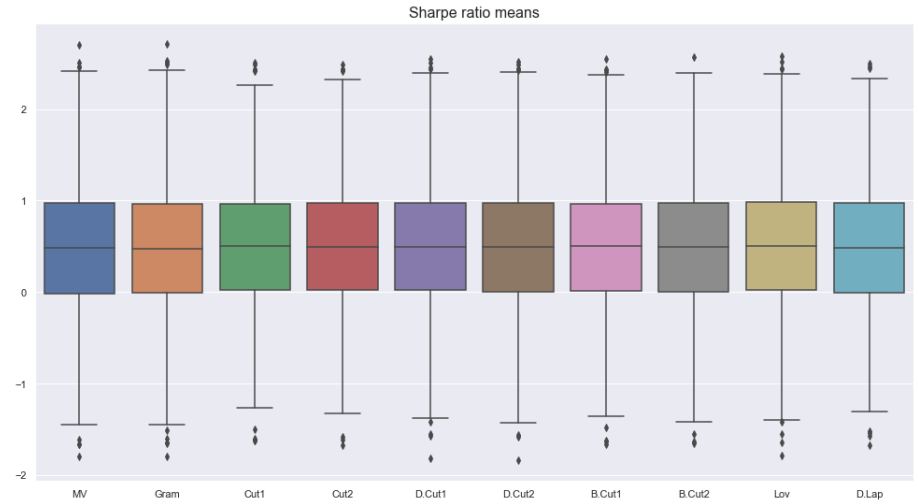


Figure 4.9: Sharpe ratio mean values for the 10 models

pected from each one of the portfolios under analysis. To better understand the real mean and variance values for this metric it was necessary to proceed first with the removal of some extreme outliers, points that showed a distorted trend for this metric in some of the portfolios considered. In this regard, mean-variance models were the models suffering more for the presence of strong outliers and as a consequence they received the greatest improvement from this removal procedure, so it will be necessary to analyze the results on MMD statistics considering this fact.

To remove the outliers in a consistent way among all the different models we first extracted Q_1 and Q_3 quartiles, found the $IQR = Q_3 - Q_1$ and determined the

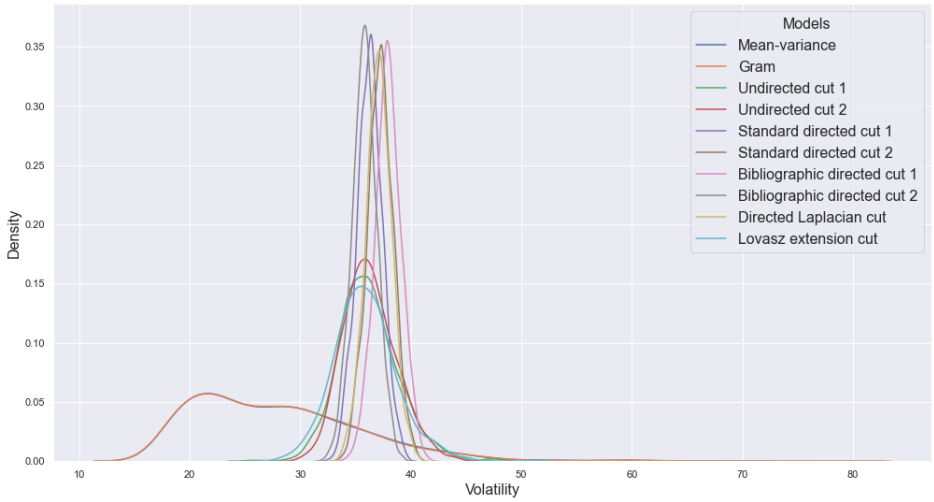


Figure 4.10: Volatility Trend over 1000 trials for all the 10 models

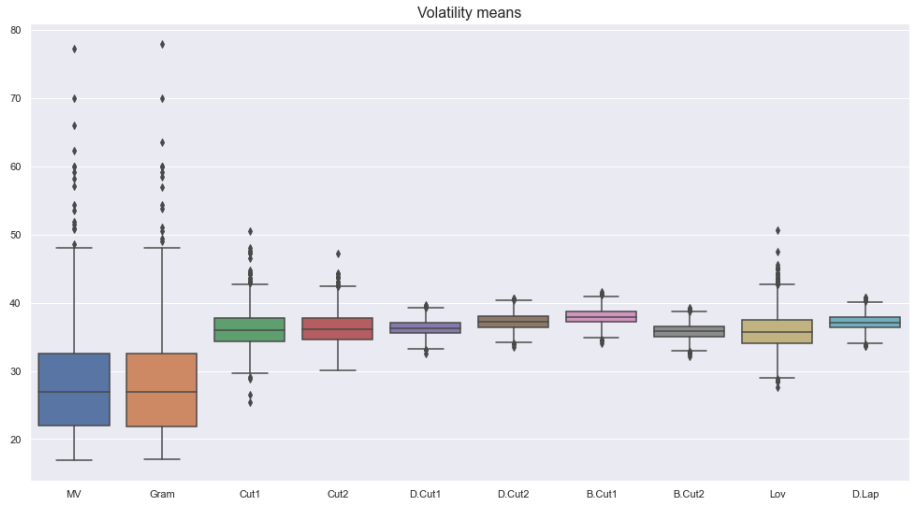


Figure 4.11: Volatility mean values for the 10 models

lowest range as $Q_1 - 1.5 \times IQR$.

Therefore, in figure 4.12 it is visible the trend in terms of maximum drawdown for the different models under analysis.

Also from figure 4.13 it is easy to see how, after the process of outlier removal, mean-variance models are the best models in terms of MDD, considering both mean values and variance ones. Among all the other clustering methods instead, standard directed clustering with the first allocation method and bibliometric directed cut with the second allocation schema seem to be the better ones, while

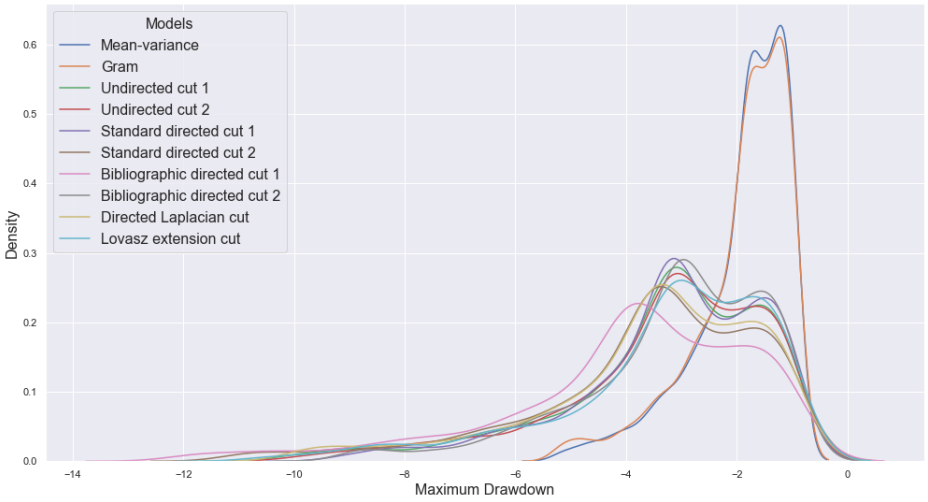


Figure 4.12: Maximum drawdown Trend over 1000 trials for all the 10 models

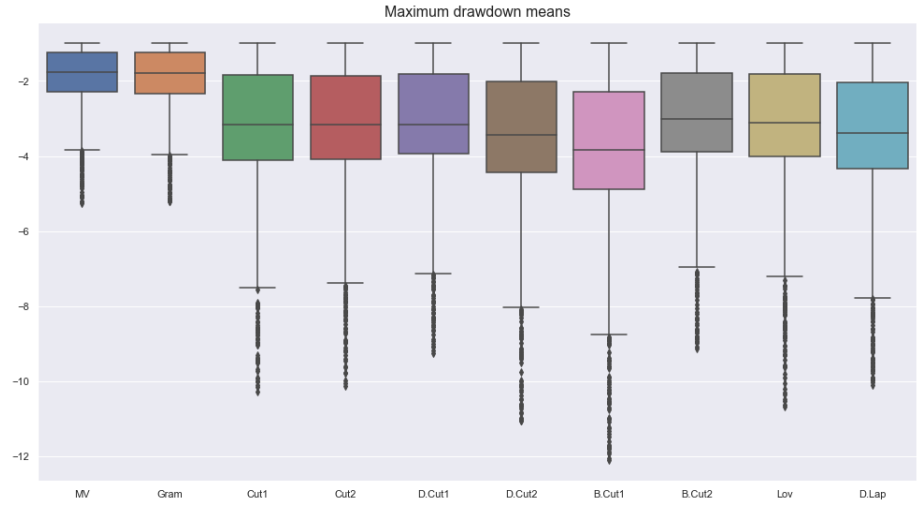


Figure 4.13: Maximum drawdown mean values for the 10 models

remaining away from the performance of benchmark mean-variance model.

Finally, we report in tables 4.6 and 4.7 the results in term of means and variances of the MDD for the different portfolios.

	Mean-Variance	Gram	Directed Laplacian	Lovász
Mean	-1.90	-1.93	-3.30	-3.52
σ^2	0.74	0.82	-3.52	-3.30

Table 4.6: Maximum drawdown statistics (1)

	Cut1	Cut2	D.Cut 1	D.Cut 2	B.Cut 1	B.Cut 2
Mean	-3.31	-3.30	-3.17	-3.60	-3.96	-3.12
σ^2	3.52	3.46	2.92	4.42	5.26	2.83

Table 4.7: Maximum drawdown statistics (2)

4.3 INTERVENTIONS ON SAMPLED DATASETS

Testing the models on a static-environment dataset is surely a good way of describing how each portfolio performs in a scenario where no sudden changes occur to the market, but to have a deeper understanding of the true performance of a certain model in a real-world scenario situation it is needed to perform interventions on the original distribution in order to simulate one or more shocks to the system.

To do this, two type of interventions were considered:

1. **hard interventions** on the system, where parental relations between assets and factors were changed drastically by deleting edges or adding them to connect previously disconnected nodes.
2. **Soft interventions**, where parental relations remained unchanged, but where noise distribution of certain assets were changed from normal ones to *Laplace distributions*.

Therefore, at each iteration where all the models were trained and tested against a regular dataset based on the causal DAG weight matrix C, other two datasets (with 252×2 points as the original test set) were sampled on the base of soft and hard interventions preformed on core nodes of the system under analysis, in this case on nodes with few connections with the whole system, that should be the main target in terms of investment of mean-variance portfolios.

In terms of Sharpe ratio values, it is possible to see how soft interventions on the system impacted harder on portfolios performances than hard interventions, especially in models such as Markovitz's one and its causal counterpart. Looking at image 4.14 it is easy to notice that the best models in terms of Sharpe ratio performance are still clustering methods such as standard directed cut (with allocation 1) and bibliometric directed cuts (with both allocation schemes), but the most interesting thing is that interventions (both soft and hard ones) had a way harder impact on benchmark Markovitz's portfolio respect to allocations obtained with clustering methods exploiting the causal structure of the data.

Looking at image 4.16, representing the relative percentage change in terms of Sharpe ratio performance between intervened systems and not intervened ones, it

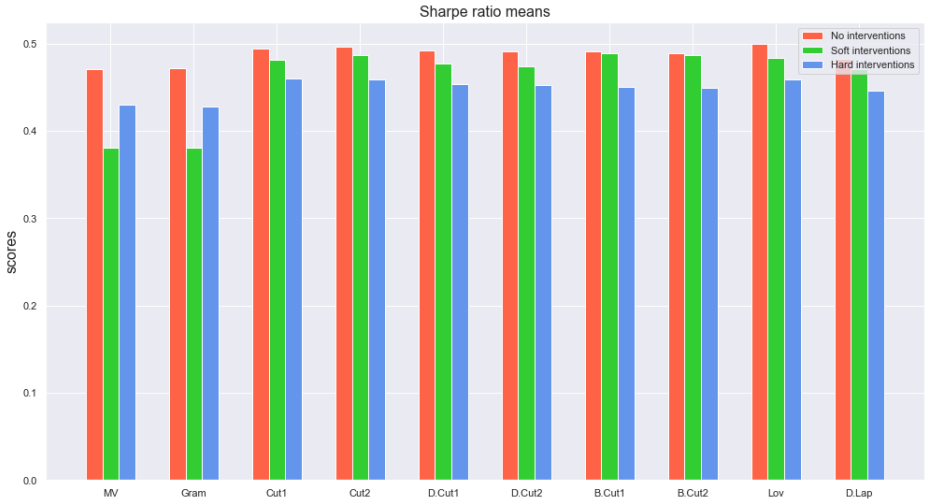


Figure 4.14: Sharpe ratio mean values obtained in a static environment (red) and after the action of soft (green) and hard (blue) interventions

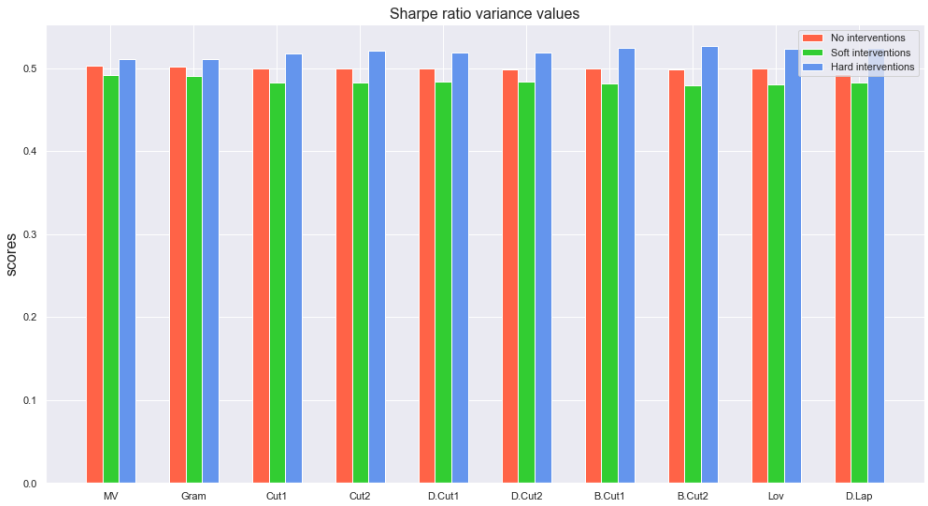


Figure 4.15: Sharpe ratio variances obtained in a static environment (red) and after the action of soft (green) and hard (blue) interventions

is easy to notice that bibliometric directed cuts method was practically unchanged in terms of Sharpe ratio values when considering the soft interventions case, where mean-variance models suffered from a huge drop in performances in this particular case. Other methods suffered more from soft interventions, but undirected cuts methods performed better than standard directed cuts, directed Laplacian model and Lovász extension model in terms of percentage Sharpe ratio loss.

When dealing with hard interventions instead, the percentage loss of all the port-

folios is almost the same. Surely mean-variance models still show the worst performances in terms of Sharpe ratio, but clustering methods are not as stable as in the soft interventions case when dealing with radical modification to the causal structure of the data.

In terms of Sharpe ratio variance, no important variations can be noticed from figure 4.15, just some slight bigger increases in the case of hard interventions for some graph clustering methods.

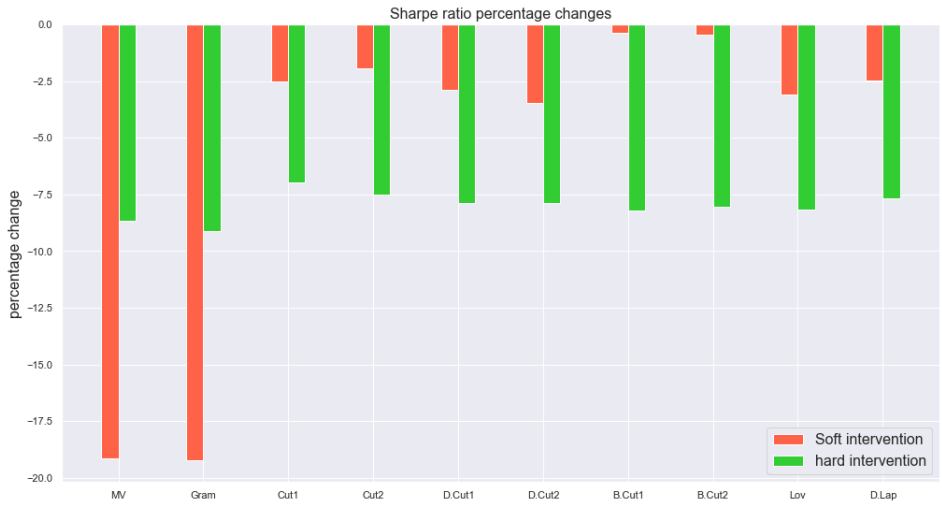


Figure 4.16: Sharpe ratio mean values percentage change in soft (red) and hard (blue) intervention cases

Figure 4.17: Volatility mean values obtained in a static environment (red) and after the action of soft (green) and hard (blue) interventions

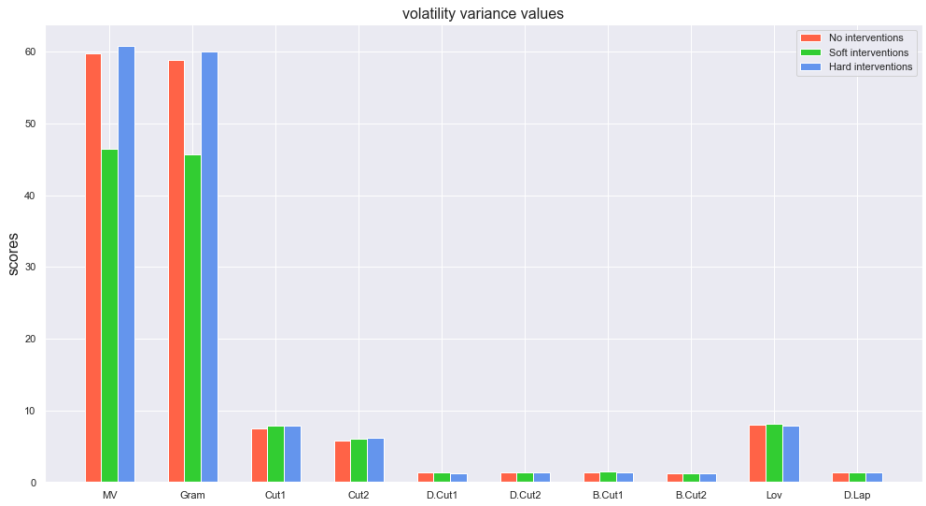


Figure 4.18: Volatility variances obtained in a static environment (red) and after the action of soft (green) and hard (blue) interventions

Going on to analyze the volatility under intervention conditions (figures 4.17 and 4.18) we can notice how basically all the clustering methods maintained the same mean and variance values for this metric both in the soft and hard intervention cases, with the exception of mean-variance portfolios, that in the soft intervened case saw a noticeable decrease in volatility variance, while at the same time increasing its mean value by a considerable margin.

This results clearer when looking at the volatility means percentage changes (fig 4.19), where the increase for mean-variance models is the only one that is worth considering.

Finally, figure 4.20 shows values for MDD's means in the three tested cases and figure 4.21 presents its percentage change in case of interventions on the system.

As saw in section 4.2.3, due to the removal of strong outliers, mean-variance portfolios obtain the best values in terms of mean MDD values in both the soft and hard interventions cases. The interesting thing is that the percentage change in performances for these models seems higher than other ones.

If considering just the soft interventions case, both bibliometric directed cut allocation methods and Lovász extension model outperform mean-variance ones when dealing with percentage changes in the case of soft interventions. In the hard intervention case, benchmark mean-variance model appears as the portfolio that looses the most in percentage, together with the first allocation method for bibliometric directed cut. Still, portfolios obtained with Lovász extension cut and with the second allocation for the bibliometric directed cut remain quite station-

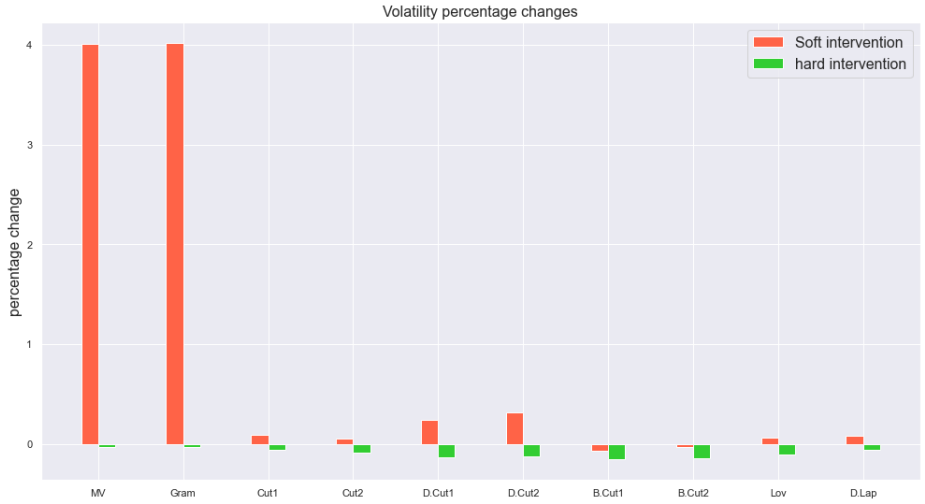


Figure 4.19: Volatility mean values percentage change in soft (red) and hard (blue) intervention cases

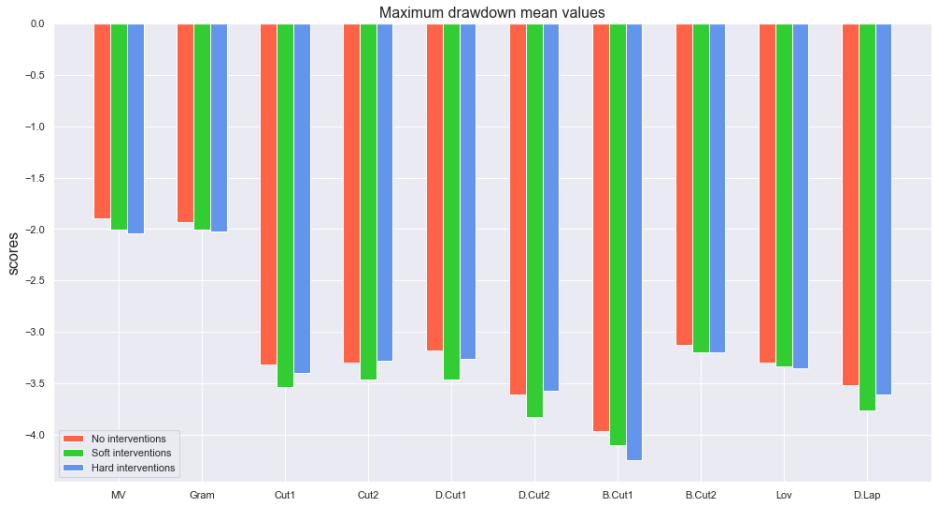


Figure 4.20: Maximum drawdown mean values obtained in a static environment (red) and after the action of soft (green) and hard (blue) interventions

ary in terms of percentage change, while undirected and standard directed cuts with allocation 2 even decrease their mean value of MDD by a small margin.

4.3.1 Final considerations on the results obtained

After analyzing the obtained results also in presence of hard and soft interventions, it is finally possible to draw some conclusion on the adequacy of the presented models to deal with asset allocation tasks.

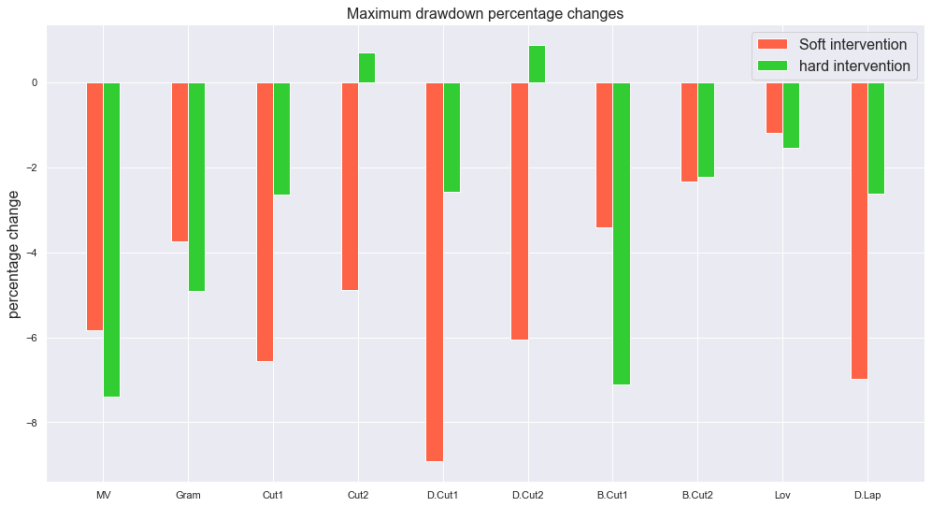


Figure 4.21: Maximum drawdown mean values percentage change in soft (red) and hard (blue) intervention cases

First, none of the models outperformed the other ones in all the circumstances and on all the metrics considered, but some showed overall good results and particularly good behavior in some intervention situations.

In terms of Sharpe ratio performances, in the static case all the models performed somewhat the same, with slight lower values for mean-variance ones. At the same time, in case of interventions on the system, some portfolios (such as Markovitz's and Gram matrix one) showed drastic percentage decrease in mean performances, where bibliometric directed cut method (with both allocation schemes considered) resulted as nearly untouched by soft intervention distributions.

From a volatility point of view, mean-variance models showed a strange behavior, obtaining the lowest values among all the portfolios while at the same time having the most flat distribution of them all, with high σ^2 values. Since Sharpe ratio deals with the maximization of expected returns while minimizing volatility, this could mean that mean-variance portfolios have serious problem in guaranteeing stable expected returns. Graph clustering methods instead, showed a more peaked trend for volatility values and as a consequence their expected return should be more stable as well.

Lastly, MDD values are the harder ones to correctly read, since outliers removal procedure impacted more on mean-variance models, showing the most swinging nature among all the portfolios and suffering more from the presence of extreme points.

Still, in these conditions mean-variance portfolios showed a good behavior in terms of MDD mean values, while, when dealing with percentage changes in interven-

tion conditions, are outperformed by other models, in particular Lovász extension cut and bibliometric directed cut methods, having a more stable nature both in hard and soft interventions cases.

It is not possible to elect an overall best model, but surely bibliometric directed cut portfolios showed to be the most stable in all of the metrics considered, while at the same time maintaining good Sharpe ratio and volatility values, guaranteeing a more stable portfolio than mean-variance ones, with however some flaws in terms of MDD. Also, it outperformed benchmark undirected cut methods, probably due to its symmetrization being able to account for edge direction and accounting for the presence of non-investible financial factors F.

5

Conclusion

After analyzing all the models presented in chapter 3, in both static and intervened conditions and considering metrics highlighting their performance under different points of view, it is finally possible to draw some conclusions on the utility of causal information in asset allocation tasks.

As already stated at the end of chapter 4, it is not possible to elect an overall best model, since all the different portfolios produced had their pros and their cons: mean-variance models, for example, were the worst performing models in terms of Sharpe ratio and had the worst values in volatility variance, but were better in terms of maximum drawdown. On the other side, outliers removal procedure favoured more these models since they had a way higher number of extreme points shifting a lot this metric's value. In the end, it is not possible to affirm with certainty that mean-variance models are worse than graph clustering ones, but it is surely possible to say that working on graph-structured data brought some important modifications that changed drastically portfolio's compositions, increasing their performance in some important aspects such as volatility and expected return stability.

However, the most important thing noticeable from tests performed under intervention conditions is the fact that, among graph clustering methods, there is a clear difference in stability between undirected methods and directed ones, where certain directed methods (such as bibliometric directed cut) showed to be more shock-resistant than undirected ones (not considering the causal structure underlying the data). It seems that causal information really played a role in making directed clustering methods more stable in situations where a sudden modification to the system occurs.

Also, in real-world scenarios, where a greater number of assets is considered, computational complexity is an issue to account for and directed methods, considering sparser weight matrices, could be less time-consuming than undirected ones in producing the optimized allocation vector **w**.

To continue this work it should be necessary to focus more on the improvement of methods that showed good results in previous tests (such as bibliometric directed cut and Lovász extension cut) with a further analysis on the optimal number of clusters needed and on different ways to allocate the budget B onto various clusters, and of course a more intense focus should be posed on the problem of inference of the causal structure, thing that if not done correctly could ruin the improvements obtained with these methods.

Ultimately, it is safe to say that causal information can have a major impact in asset allocation tasks, at least when considering portfolio's stability in times of global market instability.

References

[1] S. Perrin and T. Roncalli, "Machine Learning Optimization Algorithms & Portfolio Allocation," Sep. 2019, arXiv:1909.10233 [q-fin, stat]. [Online]. Available: http://arxiv.org/abs/1909.10233

[2] S. Theodoridis, "Probabilistic Graphical Models: Part I," in *Machine Learning*. Elsevier, 2020, pp. 771–820. [Online]. Available: https://linkinghub.elsevier.com/retrieve/pii/B9780128188033000271

[3] M. J. Vowels, N. C. Camgoz, and R. Bowden, "D'ya like DAGs? A Survey on Structure Learning and Causal Discovery," Mar. 2021, arXiv:2103.02582 [cs, stat]. [Online]. Available: http://arxiv.org/abs/2103.02582

[4] J.Peters, D.Janzing, and B.Scholkopf, *Elements of causal inference: foundations and learning algorithms*. MIT press, 2017.

[5] C. Uhler, G. Raskutti, P. Bühlmann, and B. Yu, "Geometry of the faithfulness assumption in causal inference," *The Annals of Statistics*, vol. 41, no. 2, Apr. 2013, arXiv:1207.0547 [math, stat]. [Online]. Available: http://arxiv.org/abs/1207.0547

[6] J.Pearl, *Causality (2nd ed.)*. Cambridge: Cambridge University Press, 2009.

[7] H. Markowitz, "Portfolio selection," *The Journal of Finance*, vol. 7, no. 1, pp. 77–91, 1952. [Online]. Available: http://www.jstor.org/stable/2975974

[8] W. Ford, "Chapter 7 - Vector and Matrix Norms," in *Numerical Linear Algebra with Applications*, W. Ford, Ed. Boston: Academic Press, 2015, pp. 119–144. [Online]. Available: https://www.sciencedirect.com/science/article/pii/B9780123944351000077

[9] B. S. Dees, L. Stankovic, A. G. Constantinides, and D. P. Mandic, "Portfolio Cuts: A Graph-Theoretic Framework to Diversification," Oct. 2019, arXiv:1910.05561 [cs, eess, math, q-fin]. [Online]. Available: http://arxiv.org/abs/1910.05561

[10] F. D. Malliaros and M. Vazirgiannis, "Clustering and community detection in directed networks: A survey," *Physics Reports*, vol. 533, no. 4, pp. 95–142, Dec. 2013. [Online]. Available: https://linkinghub.elsevier.com/retrieve/pii/S0370157313002822

[11] V. Satuluri and S. Parthasarathy, "Symmetrizations for clustering directed graphs," in *Proceedings of the 14th International Conference on Extending Database Technology - EDBT/ICDT '11*. Uppsala, Sweden: ACM Press, 2011, p. 343. [Online]. Available: http://portal.acm.org/citation.cfm?doid=1951365.1951407

[12] F. Chung, "Laplacians and the Cheeger Inequality for Directed Graphs," *Annals of Combinatorics*, vol. 9, no. 1, pp. 1–19, Apr. 2005. [Online]. Available: http://link.springer.com/10.1007/s00026-005-0237-z

[13] S. Sardellitti, S. Barbarossa, and P. Di Lorenzo, "On the Graph Fourier Transform for Directed Graphs," *IEEE Journal of Selected Topics in Signal Processing*, vol. 11, no. 6, pp. 796–811, Sep. 2017, arXiv:1601.05972 [math]. [Online]. Available: http://arxiv.org/abs/1601.05972

[14] U. von Luxburg, "A Tutorial on Spectral Clustering," Nov. 2007, arXiv:0711.0189 [cs]. [Online]. Available: http://arxiv.org/abs/0711.0189