Correlation Detective

Efficient Multivariate Correlation Discovery

Minartz, Koen

Award date:
2021

EINDHOVEN UNIVERSITY OF TECHNOLOGY
Department of Mathematics and Computer Science

Correlation Detective: Efficient Multivariate Correlation Discovery

Master Thesis

Koen Minartz

Supervisors:

dr. O. Papapetrou
dr. V. Menkovski
prof. dr. G.H.L. Fletcher

Eindhoven, Thursday 23rd September, 2021

Abstract

Correlation analysis is a key tool in the arsenal of data science practitioners. In a typical use-case, the data scientist may examine the most strongly correlated pairs of variables to find potentially interesting relationships. A useful generalization of this inquiry, called *multivariate correlation discovery*, aims to find strongly correlated sets of variables, as opposed to pairs. These sets are especially interesting when the set considered as a whole exhibits a strong degree of dependence, while the underlying pairwise correlations are insignificant. However, the combinatorial explosion of the search space prohibits the application of multivariate correlation discovery to even moderately sized datasets in practice. Existing methods either address only pairwise correlations or make restricting assumptions, limiting their general applicability. In this thesis, we propose a more broadly applicable algorithmic framework. Correlation Detective equips the user with a great amount of flexibility by supporting various query types, correlation measures and optional filters. Moreover, in contrast to earlier works, it guarantees completeness of the result set. Our evaluation shows that our methods are at least an order of magnitude faster than baseline and state-of-the-art algorithms.

Preface

In 2019, I decided to turn to the dark side and trade a bachelor program in Industrial Engineering for a master's in Data Science in Engineering. Although the unusual circumstances resulted in even more screen time than anticipated, I thoroughly enjoyed all courses and projects and remain very happy with this decision. This document contains the last part of the master program, namely a thesis project titled *Correlation Detective: Efficient Multivariate Correlation Discovery*. Producing this thesis would not have been possible without the support of many people.

I would like to thank my supervisor Odysseas Papapetrou for the outstanding guidance. The close involvement of Odysseas greatly increased the quality of this work, and I am truly amazed by his dedication and efforts. I would also like to thank Jens for the productive discussions on both of our projects. Additionally, I would like to thank Vlado Menkovski and George Fletcher for participating in the assessment committee.

Most importantly, I would like to thank my family, Evelien, and all friends in Voerendaal, Heerlen, Eindhoven and other places for their support and the fun times during the past five years and before.

Koen Minartz

Contents

Chapter 1

Introduction

The amount of data generated, collected, and stored is exploding. In 2018, the collective size of the world's data was 33 zettabytes, and this number is projected to grow to 175 ZB in 2025, corresponding to a compounded annual growth rate of 61% [51]. Consequently, data analytics will play a role of ever-increasing importance in organizations' business operations for the foreseeable future, and adopting data-driven practices is key for businesses in almost all sectors to remain competitive [37]. This thesis examines multivariate correlation analysis, a type of inquiry that can help domain experts to gather insights and make decisions in a data-driven fashion. Performing this analysis with an exhaustive algorithm would require considering all possible combinations of columns in the database, which is infeasible in practice for reasonably large datasets. Therefore, efficient methods are necessary to perform multivariate correlation analysis at scale. This study proposes an algorithmic framework to discover the most interestingly correlated variables orders of magnitude faster and more effectively than existing methods.

1.1 Motivation

Pairwise and Multivariate Correlation Analysis

Correlation analysis is a widely used tool in data science to explore the data, get insights into relationships between the observed variables and discover new phenomena. The following highlighted applications illustrate the relevance of pairwise correlation analysis for a diverse range of fields:

- Neuroscientists interpret strong correlations between the activity levels measured within two brain regions as an indication that the regions are anatomically or functionally related [7].
- In genetics, scientists use correlation analysis to discover and express relationships between genetic disorders and their underlying cause. An exciting initiative in this field is the SPARK project, collecting genetic information of thousands of individuals affected by autism [18]. This resulted in a list of symptoms and correlating genes [19].
- In finance, correlations between the historical returns of a set of assets are used to find portfolios that are Pareto-optimal with respect to their risk and expected return [36].
- In Computer Science, the interpretation of correlations as a generalization of functional dependencies resulted in a novel secondary indexing mechanism to accelerate query processing in relational databases [59].

Multivariate correlation measures are generalizations of pairwise correlation measures that aim to express relationships among an arbitrary amount of variables, typically stored as high-dimensional

vectors. Multivariate correlation analysis allows for the discovery of phenomena that would go unnoticed in bivariate analyses, and has therefore gained the attention of researchers in various domains. Correlations of triplets of fMRI time series led to new ideas on how different parts of the brain interact when executing audiovisual tasks, such as watching a cartoon [2, 3]. For instance, the activity of the left middle frontal region was found to have a high correlation with the average activity of the right superior frontal and left inferior frontal regions while the brain was processing audiovisual stimulus. This suggests that the left middle frontal has an integrative role of assimilating information from the other two regions. Climate Scientists used tripoles, which are correlations among three time series, to express relations of sea-level pressure observed at distant locations on the globe [34]. Figure 1.1 illustrates this finding. Multivariate correlation analysis is also shown to be relevant for applications in genetics, where the *interaction* between gene loci plays a crucial role in the expression of complex traits [9, 40, 60]. Similarly, in medicine the *combination* of hereditary predisposition and lifestyle factors has been shown to strongly correlate to the risk of developing dementia [33].

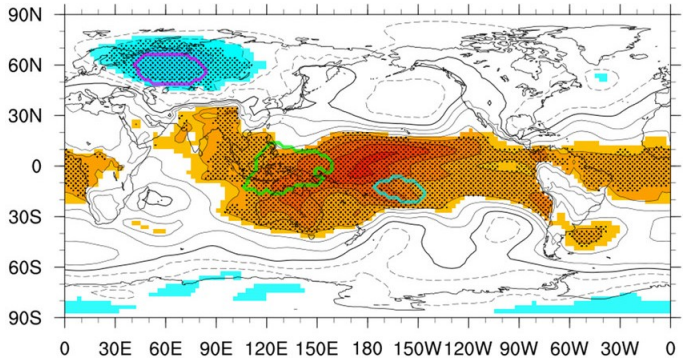


Figure 1.1: A weather phenomenon discovered by multivariate correlation analysis: the air pressure over the West Siberian Plain (magenta outline) is strongly negatively correlated to the aggregated pressure levels over Darwin, Australia (green outline) and Tahiti (cyan outline). Figure by Liess et al., 2017 [34].

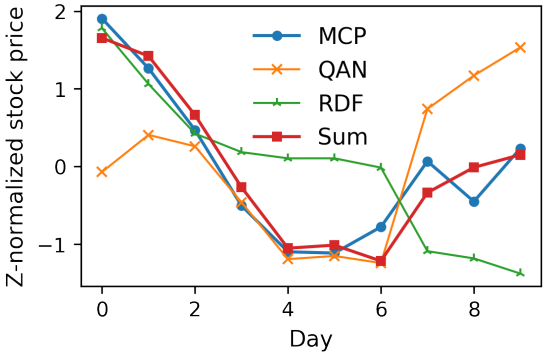
As a result of the increasing interest in multivariate correlation analysis, several correlation measures and algorithms have been researched to identify strongly correlated sets of vectors. Agrawal et al. proposed tripoles [2]. A triplet of vectors $\mathbf{y}, \mathbf{x_1}, \mathbf{x_2}$ is considered to be an interesting tripole if $\mathbf{y}$ exhibits a strong Pearson correlation with $\mathbf{x_1} + \mathbf{x_2}$ *and* if this correlation is significantly stronger than the pairwise correlations in the triplet. A follow-up work proposed multipoles [3]. Multipoles are not a generalization of tripoles. Instead, the multipole measure assesses how close an arbitrarily-sized set of vectors is to being linearly dependent. Canonical Correlation Analysis (CCA) takes as input two sets of vectors, and finds a projection of each set such that the Pearson correlation between the two projections is maximized [24]. Complementary to the above correlation measures, which are all linear in nature, non-linear multivariate correlation measures are also actively researched. Total Correlation originates from information theory and expresses the amount of information that is shared between vectors [58]. Due to its flexibility, various measures based on Total Correlation have been proposed to express interesting relationships between variables [45, 46, 60]. All above mentioned multivariate correlation measures complement each other and are useful in different contexts.

Challenges of Multivariate Correlation Discovery

Despite its usefulness, using multivariate correlation analysis to identify strongly related vectors in any reasonably-sized data set remains challenging due to the inherently large search space; the

whole powerset of the data needs to be considered in order to find all strongly correlated sets of vectors. Therefore, exhaustive search algorithms are infeasible in practice. As an example, finding all strongly correlated quadruplets of vectors in a database containing 1000 time series would require enumerating in the order of 1 trillion candidate sets. Even for this dataset, which is nowhere near 'Big Data', iterating over all candidates already requires significant computational resources.

Simultaneously, the strength of bivariate correlations does not directly indicate whether or not the pair can be part of a strongly correlated set of vectors. Consider as an example Figure 1.2, depicting daily closing prices and Pearson correlations of three stocks that are traded at the Australian Stock Exchange. For ease of exhibition, all time series in the figure are z-normalized – this implies that highly correlated time series are plotted close to each other. The absolute values of the pairwise correlations of MCP, QAN and RDF's closing prices are all around 0.5, whereas the Pearson correlation of MCP with the sum of QAN and RDF is 0.96. This simple example illustrates that comparably weakly correlated pairs of vectors may still be part of a very strongly correlated set of vectors.



	MCP	QAN	RDF	Sum
MCP	1	0.46	0.53	0.96
QAN	0.46	1	-0.47	0.52
RDF	0.53	-0.47	1	0.52
Sum	0.96	0.52	0.52	1

Figure 1.2: Z-normalized closing prices and correlations of McPherson's Ltd (MCP), Qantas (QAN) and Redflex Holdings Limited (RDF), as well as the z-normalized sum of the latter two assets' closing prices. Figure adapted from Minartz, d'Hondt and Papapetrou, 2021 [39].

Limitations of Prior Work and Contributions

It is clear that sophisticated algorithms are needed that can drastically prune the search space in order to make the identification of strongly correlated sets of vectors feasible in practice. Typically, existing algorithms exploit one or more of the following strategies in order to reduce the amount of candidates that need to be considered:

- The algorithm addresses a correlation measure that allows pruning using the apriori property [3, 45, 60].
- The algorithm relies on a hand-crafted additional constraint (other than the desired correlation strength) to prune candidates [2, 3, 60]
- The algorithm is an *approximation* algorithm, meaning that it does not guarantee any degree of completeness of the result set [2, 3, 45].

The above-mentioned algorithms are very useful for applications where the underlying pruning strategies and assumptions do not impede the goal of the domain expert. However, in general, they constrain the flexibility of the user. Addressing this issue, this study proposes an algorithmic framework that offers the user more flexibility to get all relevant results. First, the algorithm does not rely on the apriori property of the correlation measure for its pruning power. This enables the

discovery of sets where the pairwise correlations are relatively weak, but the correlation of the set as a whole is strong, which are arguably the most interesting cases.

Second, inspired by Ockham's Razor, the algorithm allows the user to *optionally* specify additional constraints that filter unnecessarily complex vector sets from the result set. More specifically, adding a vector to an existing set is only allowed if the resulting larger set is significantly more interesting, i.e. more strongly correlated. This leads to a result set that contains interpretable results, and prevents clutter by large vector sets in which some of the vectors contribute only marginally to the strength of the relationship.

Finally, three algorithmic query types are supported, from which the user may choose the most suitable depending on the context of the application. The first is a threshold query, in which all sets that exceed a user-specified correlation threshold are retrieved. The second is a progressive query, which yields the bulk of the threshold query's results during a comparably short part of the total execution time, and continuously adds results as the algorithm proceeds. The third query is a top-k query, yielding the k strongest correlated vector sets that satisfy the user's query.

1.2 Context

This thesis is part of an overarching project titled *Correlation Detective: explainable multiple correlations for the real world*. Three master theses were started in parallel under this common denominator. The first thesis is the one you are reading. Its methodology serves as a basis for the other theses to build upon or to be adapted for application to their respective domains.

The second thesis applies the discovery of multivariate correlations to the domain of finance. Imagine that a big financial institution wants to remove all assets of which the CO_2 footprint exceeds a certain norm from its portfolio. Applying mathematical programming techniques to optimize the portfolio from scratch would require many (small) transactions and therefore lead to substantial transaction costs. Instead, the thesis applies multivariate correlation discovery to find a diverse, initial set of candidates that could potentially replace the securities that are to be removed: if candidate assets are highly correlated to the removed securities, their returns are expected to follow a similar pattern and the portfolio is expected closely follow the original portfolio in terms of risk and expected profits. Moreover, finding a diverse set of candidate assets allows domain experts to consider qualitative aspects such as geographic or industry diversification. The thesis resulted in a framework that significantly outperforms benchmarks in terms of the tracking error of the portfolio with the replaced assets.

The third thesis focuses on the discovery of multivariate correlations over sliding windows in streaming data. In many applications related to time series, correlations may be non-stationary due to changes in the data generating process. One such application is found in the field of neuroscience: even in resting-state fMRI scans, where subjects are not performing any tasks, correlations between activity levels in different parts of the brain are generally non-stationary and fluctuate over time [23]. Another example domain is finance, where the actions of an extravagant CEO appearing in a podcast may abruptly change the popularity of a stock. Efficient maintenance of the result set is crucial to enable the exploitation of multivariate correlation discovery in applications where results must be available in real-time, such as flash trading [52] and live neurofeedback therapy [28, 38, 62].

The combined efforts of the third thesis and this thesis led to a submission under review for the 48^{th} International Conference on Very Large Data Bases [55], of which the technical report is available via [39].

1.3 Outline

The remainder of this document is structured as follows: in Chapter 2, we will introduce the most relevant correlation measures and formalize the problem statement. Chapter 3 discusses related work in the area of (multivariate) correlation discovery. In Chapter 4, theory and algorithms are presented to efficiently discover strong multivariate correlations. In Chapter 5, the methodology is evaluated on real-life dataset, and the results are presented and interpreted. Finally, Chapter 6 concludes the study and proposes directions for future research.

Chapter 2

Preliminaries

2.1 Correlation Measures

This section presents correlation measures that are essential to our methodology. Other correlation measures that are interesting, but not directly used in this work, are presented in Chapter 3.

Pearson's ρ

Pearson's ρ is a ubiquitously used correlation measure. The Pearson correlation is a measure that expresses the degree of linear dependence between a vector $\mathbf{x} \in \mathbb{R}^d$ on the left hand side, and a vector $\mathbf{y} \in \mathbb{R}^d$ on the right hand side; $\rho = 1$ is interpreted as a perfect positive linear relationship between both vectors, whereas $\rho = -1$ indicates a perfect negative linear correlation. Note that $\rho = 0$ only indicates that there is no linear relation between the vectors: ρ may be equal to zero even if there is a perfect nonlinear dependence between $\mathbf{x}$ and $\mathbf{y}$.

With $\mu(\mathbf{x})$ and $\sigma(\mathbf{x})$ denoting the mean and standard deviation of the elements in $\mathbf{x}$ respectively, ρ is defined as:

$$\rho(\mathbf{x}, \mathbf{y}) = \frac{1}{d} \frac{\sum_{i=1}^{d} (x_i - \mu(\mathbf{x})) \cdot (y_i - \mu(\mathbf{y}))}{\sigma(\mathbf{x}) \cdot \sigma(\mathbf{y})} \tag{2.1}$$

In the case the vectors $\mathbf{x}$ and $\mathbf{y}$ are z-normalized, meaning the vectors have zero mean and unit standard deviation, the equation simplifies to:

$$\rho(\mathbf{x}, \mathbf{y}) = \frac{1}{d} \sum_{i=1}^{d} x_i \cdot y_i = \frac{1}{d} \mathbf{x}^T \mathbf{y} \tag{2.2}$$

Where $\mathbf{x}^T$ denotes the transpose of $\mathbf{x}$. One interesting observation about the Pearson correlation between two z-normalized vectors is that it is directly related to the Euclidean distance between these vectors [43, 61]. In particular, with $d^2(\mathbf{x}, \mathbf{y})$ denoting the squared Euclidean distance between $\mathbf{x}$ and $\mathbf{y}$, for z-normalized $\mathbf{x}$ and $\mathbf{y}$ we have:

$$\rho(\mathbf{x}, \mathbf{y}) = 1 - \frac{1}{2d} d^2(\mathbf{x}, \mathbf{y}) \tag{2.3}$$

Moreover, ρ of two z-normalized vectors $\mathbf{x}$ and $\mathbf{y}$ is equal to the cosine of the angle $\theta_{\mathbf{x},\mathbf{y}}$ between those vectors:

$$\rho(\mathbf{x}, \mathbf{y}) = \frac{1}{d} \mathbf{x}^T \mathbf{y} = \frac{\mathbf{x}^T \mathbf{y}}{\sqrt{d} \cdot \sqrt{d}} = \frac{\mathbf{x}^T \mathbf{y}}{||\mathbf{x}||_2 \cdot ||\mathbf{y}||_2} = \cos(\theta_{\mathbf{x},\mathbf{y}}) \tag{2.4}$$

As such, a high Pearson correlation of z-normalized vectors indicates that the angle between these vectors is small. The relevance of the relation between ρ, Euclidean distance and angles will

become apparent in Chapter 4, where the pruning mechanism of the Correlation Detective is explained.

Tripole

In 2017, Agrawal et al. proposed the tripole correlation measure [2]. The tripole correlation extends Pearson and is based on the observation that a vector, say $\mathbf{y}$, may be more strongly correlated to the sum of two vectors $\mathbf{x_1}$ and $\mathbf{x_2}$ than it is to either of the two – recall Figure 1.2. Based on this observation, the tripole measure is defined as follows:

$$\text{tripole}\left(\mathbf{y}, \{\mathbf{x_1}, \mathbf{x_2}\}\right) = \rho(\hat{\mathbf{y}}, \hat{\mathbf{x}}_\mathbf{1} + \hat{\mathbf{x}}_\mathbf{2}) \tag{2.5}$$

where $\hat{\mathbf{v}}$ denotes $\mathbf{v}$ after z-normalization. The goal of [2] was to discover triplets that have a significantly stronger correlation than any of the pairwise correlations between the vectors in the triplet. In addition, redundancy of the result set is prevented by ensuring that no two results that are very similar are returned; that is, no two distinct triplets are in the result set such that each variable in one triplet is highly correlated to another variable in the other. A naive approach would require enumerating over all triplets of columns in the database and thus suffers from cubic complexity. Therefore, an exhaustive algorithm becomes practically infeasible for data sets containing in the order of tens of thousands of vectors .

The relevance of the comparatively simple tripole measure was demonstrated by applying the algorithms to data in the domains of climate science and neurology. As already shown in Figure 1.1, the tripole measure and algorithms proposed in [2] aided in the discovery and characterization of a connection between the climate in West-Siberia and regions in the Pacific ocean [34].

Multipole

In 2020, Agrawal and his colleagues proposed another correlation measure called multipole [3]. The multipole measure is defined as follows: let $A = \{\mathbf{a_1}, \ldots, \mathbf{a_n}\}$ be a set containing z-normalized column vectors in the d-dimensional space, and let $\mathbf{A}$ be the matrix $[\mathbf{a_1} \ldots \mathbf{a_n}]$ formed by concatenating the vectors. The multipole correlation then equals

$$\mathbf{mp}(A) = 1 - \min_{||\mathbf{w}||=1} \text{Var}(\mathbf{A}\mathbf{w}) \tag{2.6}$$

in which $\mathbf{w}$ is a vector of weights. As such, the multipole measure is defined as the solution of a constrained optimization problem, which minimizes the variance of a linear combination of the vectors in A. The measure can be interpreted as follows: if the vectors in $\mathbf{A}$ are linearly dependent, $\mathbf{A}$ is not of full rank, and as such, the smallest eigenvalue will be equal to 0 and $\mathbf{mp}(A) = 1$. If the vectors are close to, but not perfectly dependent, then the smallest eigenvalue will not equal zero, but it will be close to zero, and as such the multipole correlation is high.

Although this measure uses the notion of linear dependence to handle an arbitrary amount of vectors, it is not a generalization of the tripole measure. This follows from the fact that the multipole measure is one-sided: there is no distinction between a left and right hand side, whereas the tripole pattern has one vector $\mathbf{y}$ that could be interpreted as dependent variable, depending on the context. Moreover, the multipole measure assigns weights to each vector that maximize the correlation value, whereas the tripole measure uses equal-weighted combinations of vectors.

2.2 Problem Definition

In this section, the formal definition of the problem statement addressed in this project is presented. To this end, we first define the general multivariate correlation discovery problem, and subsequently present which instances of the general problem definition are addressed in this study.

General Multivariate Correlation Discovery Problem

The goal of this work is to discover interestingly correlated subsets of a set of vectors $\mathcal{V} = \left\{\mathbf{v_i} \in \mathbb{R}^d\right\}_{i=1}^{n}$, that satisfy all constraints that are provided by the user. In order to formally define the problem, let us first define a generic correlation measure:

Definition 2.1 (**Correlation Measure**). Let $X = \left\{\mathbf{x_i} \in \mathbb{R}^d\right\}_{i=1}^{l}$ and $Y = \left\{\mathbf{y_i} \in \mathbb{R}^d\right\}_{i=1}^{r}$ be two arbitrarily-sized sets of d-dimensional vectors. A correlation measure $Corr(X, Y)$ maps to a real number that indicates the degree of correlatedness of the vectors in X and Y. There also exist correlation measures that only accept one set of vectors as argument as opposed to two; we refer to these measures as *one-sided*, as opposed to *two-sided*. To avoid textual clutter from distinguishing the two cases, we will use $Corr$ for both in the remainder of this work, with the understanding that $Corr(X, Y)$ is simply $Corr(X)$ in case the measure accepts only a single set of vectors.

The objective of the general multivariate correlation discovery problem is to find all vector sets of which the correlation is in a desired range R, and that satisfy all additional filters provided by the user. Formally, the general problem statement is defined as follows:

Definition 2.2 (**Multivariate Correlation Discovery Problem**). Given a data set $\mathcal{V}$, a user-defined correlation measure $Corr$, an interesting correlation value range R, and a possibly empty set Q of additional filtering predicates that serve as constraints on (X, Y), the objective of the multivariate correlation discovery problem is to find the result set $\mathcal{R}$ as defined in Equation 2.7:

$$\mathcal{R} = \{(X, Y) \mid X, Y \subseteq \mathcal{V} \wedge Corr(X, Y) \in R \wedge \forall_{q_i \in Q} : q_i(X, Y)\} \tag{2.7}$$

Intuitively, the objective of the Multivariate Correlation Discovery problem is to find all subsets X, Y of $\mathcal{V}$ such that $Corr(X, Y)$ is in the set of interesting correlation values R. Additionally, if specified, user-defined constraints on X, Y must be satisfied as well; examples of such constraints will be presented later in this section.

Correlation Measures

This research addresses two multivariate correlation measures. Both of these operate on z-normalized vectors: each vector's elements have zero mean and unit standard deviation. The first measure is the natural extension of the tripole correlation measure proposed in [2] (See also Section 2.1). The multiple Pearson correlation measure, abbreviated as **mc**, extends the tripole measure from ternary to n-ary correlations:

$$\mathbf{mc}(X, Y) = \rho\left(\frac{\sum_{\mathbf{x} \in X} \mathbf{x}}{|X|}, \frac{\sum_{\mathbf{y} \in Y} \mathbf{y}}{|Y|}\right) \tag{2.8}$$

Note that, as **mc** accepts two vector sets as input, it is a two-sided measure. In this work, we will focus on simple equally-weighted summations of vectors, but the theory and algorithms that are presented can straightforwardly be extended to weighted summations of vectors in general.

In addition to **mc**, the multipole measure (abbreviated as **mp**) is also addressed. The definition and properties of this measure can be found in Section 2.1.

Query Types

Depending on the application, the user may be interested in finding all strong correlations, a subset of those, or the set of strongest correlations that can be found in the data. To accommodate for this, our framework supports three complementary query types, that are special cases of the multivariate correlation discovery problem as defined in Definition 2.2.

- *Threshold query*: for a user-chosen correlation measure $Corr$, correlation threshold τ, and set of additional constraints Q, find

$$\mathcal{R} = \{(X, Y) \mid X, Y \subseteq \mathcal{V} \wedge Corr(X, Y) \geq \tau \wedge \forall_{q_i \in Q} : q_i(X, Y)\}$$

  Intuitively, the threshold query finds all X, Y that are correlated stronger that the threshold τ specified by the user, and also satisfy all constraints in Q.

- *Progressive query*: the goal of the progressive query is to progressively yield partial results of the threshold query as the execution proceeds. The algorithm for executing this query should be such that the bulk of the results is found during a comparatively short initial part of the total run time, and a complete result set is found when the algorithm is allowed to run until completion. Note that the progressive query can also be used as an approximation query by simply terminating the execution at a user-set time limit, or when a user-specified amount of results has been discovered already.

- *Top-k query*: for a user-chosen correlation measure $Corr$, and an integer parameter k, a set $\mathcal{R}$ of cardinality k should be returned for which

$$\begin{aligned}\mathcal{R} = \{(X, Y) \mid X, Y \subseteq \mathcal{V} \wedge \forall_{q_i \in Q} : q_i(X, Y) \\ \wedge \left(\neg\exists_{(X', Y') \notin \mathcal{R}} : X', Y' \subseteq \mathcal{V} \wedge Corr(X', Y') > Corr(X, Y) \wedge \forall_{q_i \in Q} : q_i(X', Y')\right)\}\end{aligned}$$

  Intuitively, the top-k query finds the k strongest correlated X, Y that also satisfy all additional constraints in Q.

Additional Constraints

Users may also want to specify a set of additional constraints, which are linked to the application scenario. Typically, these constraints relate to the targeted diversity and significance of the returned solutions. In the remainder of this work, we will assume that the user is not interested in trivial solutions (X, Y), where X and Y have a non-empty intersection, and that the user wants to limit the size of the retrieved vector sets to a maximum cardinality. This corresponds to the following predicates q_i:

- *Non-overlapping sets constraint:* to prevent redundant answers that give the user no significant insights, we will assume that all $(X, Y) \in \mathcal{R}$ should be such that there are no vectors in X that are also in Y. More specifically,

$$q_{\text{non-overlapping}}(X, Y) \coloneqq (X \cap Y = \emptyset)$$

- *Maximum cardinality constraint:* abiding by Ockham's Razor, our focus is on the discovery of interpretable multivariate correlations that are not overly complex. As such, we assume that the user defines parameters $l_{\max}$ and $r_{\max}$, such that for all $(X, Y) \in \mathcal{R}$, $|X| \leq l_{\max}$ and $|Y| \leq r_{\max}$. More formally:

$$q_{\text{max-cardinality}}(X, Y) \coloneqq (|X| \leq l_{\max} \wedge |Y| \leq r_{\max})$$

Furthermore, we consider two optional constraints that can be added to the query, but other proposed constraints (for example the weak-correlated feature subset constraint found in [60]) could easily be integrated in the methodology. Correlation Detective not only supports these optional constraints, but can actively exploit their restrictive nature on the result set to prune the search space.

- *Irreducibility constraint*: this constraint is again motivated by Ockham's razor. If the user decides to add this constraint to the query, all $(X, Y) \in \mathcal{R}$ should be such that removing any number of vectors from X, Y or both results in a correlation that is not in the result set $\mathcal{R}$. More formally:

$$q_{\text{irreducibility}}(X, Y) \coloneqq (\forall_{X' \subseteq X, Y' \subseteq Y} : (X', Y') \notin \mathcal{R} \vee (X', Y') = (X, Y))$$

  This constraint prioritizes smaller answers (smaller sets X and Y), which are easier to interpret.

- *Minimum jump constraint*: This constraint, as proposed in [2, 3], ensures that all vectors in X and Y contribute significantly to the value of $Corr(X, Y)$. Conceptually, this corresponds to avoiding answers in which there are vectors that add very little value to the strength of the multivariate relation. For a significance threshold δ, the minimum jump constraint is defined as follows:

$$q_{\text{minimum-jump}}(X, Y) \coloneqq (\forall_{X' \subseteq X, Y' \subseteq Y} : Corr(X', Y') + \delta < Corr(X, Y) \vee (X', Y') = (X, Y))$$

The minimum jump constraint applies to all query types. On the other hand, the irreducibility constraint is only relevant for threshold and progressive queries, since for the top-k query, the result set is ill-defined as it depends on the order in which results are considered. Take as a contrived example a top-1 query, and some vectors $\mathbf{a}, \mathbf{b}, \mathbf{c}$ for which $Corr(\{\mathbf{a}\}, \{\mathbf{b}\}) = 0.8$ and $Corr(\{\mathbf{a}\}, \{\mathbf{b}, \mathbf{c}\}) = 0.9$. If $(\{\mathbf{a}\}, \{\mathbf{b}\})$ is evaluated first, it will be put in the result set, and as such, when $(\{\mathbf{a}\}, \{\mathbf{b}, \mathbf{c}\})$ is subsequently evaluated, it should be discarded as the irreducibility constraint is violated. However, if $(\{\mathbf{a}\}, \{\mathbf{b}, \mathbf{c}\})$ is evaluated first, it is put in $\mathcal{R}$, and since $(\{\mathbf{a}\}, \{\mathbf{b}\})$ has a lower correlation, it is discarded. To avoid such ambiguities, we do not consider the irreducibility constraint in combination with a top-k query.

Finally, to facilitate ensuing discussions, we define the notion of a correlation pattern:

Definition 2.3 (**Correlation Pattern**)**.** A correlation pattern consists of the correlation measure $Corr$ and maximum left hand side and right hand side cardinalities $l_{\max}$ and $r_{\max}$, and is denoted as $Corr(l_{\max}, r_{\max})$. For example, a correlation pattern with **mc** correlation and maximum cardinalities $1, 2$ is denoted as $\mathbf{mc}(1, 2)$, and the multipole correlation with maximum cardinality 5 is denoted as $\mathbf{mp}(5)$.

Chapter 3

Related Work

In this chapter, we discuss the most relevant works related to the efficient discovery of strong correlations. Continuing the discussion of Chapter 2, we start by introducing additional multivariate correlation measures that, although very useful, are left out of the scope of this work. Then, an overview for methods addressing the discovery of strong pairwise correlations is presented, and we discuss why these methods cannot be straightforwardly adapted to the case of multivariate correlation discovery. Subsequently, we extend the discussion to works addressing the discovery of multivariate correlations. Finally, we briefly touch upon some works addressing tangent problems.

3.1 Multivariate Correlation Measures

Canonical Correlation Analysis

Canonical Correlation Analysis (CCA) [8] is a generalization of the ordinary pairwise Pearson coefficient. This measure accepts two sets of vectors $X = \{\mathbf{x_1}, \ldots, \mathbf{x_{l_{max}}}\}$ and $Y = \{\mathbf{y_1}, \ldots, \mathbf{y_{r_{max}}}\}$ as input. The CCA-correlation is then defined as follows:

$$\text{CCA}(X, Y) = \max_{||\mathbf{u}||=1, ||\mathbf{w}||=1} \rho(\mathbf{Xu}, \mathbf{Yw}) \tag{3.1}$$

where $\mathbf{X} = [\mathbf{x_1} \ldots \mathbf{x_{l_{max}}}]$ and $\mathbf{Y} = [\mathbf{y_1} \ldots \mathbf{y_{r_{max}}}]$ are matrices and $\mathbf{u}$ and $\mathbf{w}$ are weight vectors. The interpretation of CCA is to find the linear combinations of the vectors in X and Y, such that the resulting vectors are as strongly linearly correlated as possible. The solution to this optimization problem is the square root of the largest eigenvalue of the matrix $\mathbf{K}$:

$$\mathbf{K} = \mathbf{\Sigma_{XX}}^{-\frac{1}{2}} \mathbf{\Sigma_{XY}} \mathbf{\Sigma_{YY}}^{-1} \mathbf{\Sigma_{YX}} \mathbf{\Sigma_{XX}}^{-\frac{1}{2}} \tag{3.2}$$

where $\mathbf{\Sigma_{XY}}$ is the covariance matrix between the vectors in X and Y, and all other matrices are analogously defined. In the case of z-normalized vectors, the covariance matrix is equal to the correlation matrix. The CCA measure may be interpreted as the degree to which a change in one set of variables (X) is linearly related to change in another set of variables (Y). Moreover, as an extension to classical CCA, kernel methods have been proposed to capture non-linear relationships [20]. Instead of finding the linear mappings $\mathbf{u}$ and $\mathbf{w}$ that maximize the correlation $\rho(\mathbf{Xu}, \mathbf{Yw})$, functions f_X and f_Y are discovered such that $\rho(f_X(X), f_Y(Y))$ is maximal. This is analogous to, e.g., Support Vector Machines, a class of linear models that can be extended to non-linearity by using the kernel trick [27].

CCA has been applied in various domains, such as neuroscience [56] and finance [5]. However, CCA is typically applied to all available variables, for which the user has to decide which variables are put in the independent set, and which ones in the dependent set. In contrast, our goal is to find a diverse result set of highly correlated variables, instead of finding a single projection that

maximizes a correlation value. Moreover, the weight vectors of the optimal projection may consist of many elements that are small in absolute value. For large datasets, this can be difficult, if not impossible, to interpret by a domain expert.

To make sure the output of the CCA measure is interpretable, the weight vectors $\mathbf{u}$ and $\mathbf{w}$ should be sparse. One adaptation of CCA that aims to address this is sparse CCA [25], employing l_1-regularization to return sparse weight vectors. Sparse CCA succeeds in returning only a subset of the dimensions, being those that are associated with a nonzero weight in the solution to the CCA optimization problem. However, only one single subset of vectors is returned, whereas in a large, heterogeneous set of data, multiple interesting relations are likely to exist. Moreover, this methodology still requires the user to divide the data into two partitions. Although this is trivial in some contexts where there is a natural partitioning (e.g., natural language data in both French and English, where the French text is the left hand side, and the English text is the right hand side), in general we consider this division to be part of the solution to the user's inquiry, and not as user input.

Mutual Information and Total Correlation

Mutual information is a correlation measure defined on pairs of vectors that is based on the notion of entropy. For a random variable X, the entropy H is defined as in [13]:

$$H(X) = \mathbb{E}\left[-\log p(x)\right]$$

The Mutual Information correlation measure $I(X, Y)$ is defined as the reduction in entropy that is achieved by considering the joint distribution $p(x, y)$ of two random variables X and Y, instead of their marginal distributions. It can be computed as follows [13]:

$$\begin{aligned} I(X,Y) &= \mathbb{E}\left[\log \frac{p(x,y)}{p(x)p(y)}\right] \\ &= H(X) + H(Y) - H(X,Y) \end{aligned}$$

Total correlation, sometimes called multi-information, is the natural extension of mutual information to patterns containing more than two vectors. Its computation is analogous to that of Mutual Information:

$$TC(X_1, X_2, \ldots, X_k) = \sum_{i=1}^{k} H(X_i) - H(X_1, X_2, \ldots, X_k)$$

An advantage of Mutual Information and Total Correlation over Pearson and other linear measures is that it expresses how much information is shared among random variables. As such, also non-linear relationships between variables can be captured. However, for samples of continuous random variables, estimating these quantities in practice requires discretizing the distribution, whereas linear measures do not face this issue.

3.2 Pairwise Correlation Discovery Algorithms

Exploratory Data Analysis (EDA) is a crucial step in any data science task. Examining the correlations between observed variables can lead to first insights about the relationships in the data which can be used in subsequent model-design stages. Consequently, many methods have been

proposed to address the problem of finding strongly correlated pairs in large data sets, where naive exhaustive approaches suffer from computational limitations.

Zhu and Shasha [61] proposed the StatStream system for discovering pairwise correlations higher than a threshold τ in streaming data. Their methodology leverages the equivalence between Pearson correlation and Euclidean distance (see Equation 2.3). Their main approach consists of reducing the vectors' dimensionality using Discrete Fourier Transformations (DFT). DFT preserves Euclidean distance, and the first few coefficients are typically a good approximation to the full series for somewhat well-behaved time series. Exploiting these properties, upper bounds on pairwise correlations are derived by comparing vectors of much lower dimensionality. Additionally, the DFT-transformed vectors are hashed to a low-dimensional grid designed such that only vectors in adjacent hypercubes need to be considered, without losing completeness of the result set.

Building on this work, Mueen et al. [43] also exploit DFTs to reduce computational complexity. Rather than indexing vectors in a grid, pairwise correlations are bounded by (1) bounding the approximation error resulting from using only the first coefficients of the DFT-transformed vectors, and (2) a dynamic programming approach based on the (reverse) triangle inequality to reduce the number of comparisons between vectors. Furthermore, the data is partitioned into batches such that likely correlated time series are part of the same batch as much as possible, minimizing I/O cost.

Methods other than DFT to reduce dimensionality have also been used in data mining algorithms for time series data. Keogh et al. [30] developed the Piecewise Aggregate Approximation (PAA) method, which reduces dimensionality by approximating values in equi-width time buckets with the bucket's mean value. The reduced dimensionality makes that the series can be indexed more effectively for similarity search, of which pairwise correlation discovery is a special case. Lin et al. [35] proposed a follow-up technique called Symbolic Aggregate Approximation (SAX). This methodology relies on approximating time series with both equi-width time buckets using PAA, and subsequently equi-depth value buckets. Lower bounds on Euclidean distance between time series can be derived using the approximated time series (and thus also upper bounds on the Pearson correlation). In a similar spirit, Euclidean Locality Sensitive Hashing (LSH) is a technique to reduce the dimensionality of high-dimensional vectors by projecting them on a comparatively small set of random vectors [15, 53]. Again, the lower dimensionality vectors are indexed in a grid, with vectors ending in the same hypercube being labeled similar. This indexing approach may produce false positives and false negatives, but the grid's dimensionality and cell sizes can be tuned to trade-off computational performance, recall and precision with probabilistic guarantees.

Observe that all of the above approaches build on two strategies: (1) some form of dimensionality reduction that comes with limited approximation error on correlations between full-dimensionality vectors by instead using their reduced-dimensionality counterparts, and (2) an index or algorithm that reduces the number of comparisons that need to be made between vectors, relying on the equivalence between high Pearson correlations and proximity in Euclidean space. Strategy (1) is irrelevant to our problem statement. In Chapter 4, it will become apparent that the asymptotic complexity for the discovery of correlations involving more than two vectors is constant in the dimensionality d of each vector. This nullifies the utility of any form of dimensionality reduction. Furthermore, strategy (2) is inapplicable to our problem: recall from Figure 1.2 that a strong multivariate correlation does not necessarily correspond to proximity of the vectors in Euclidean space. One could consider the strategy of indexing all *aggregates* of vectors to discover which aggregates are close to each other (i.e., have a high **mc** correlation). However, this would not scale due to the high time and space complexity of constructing such an index. Moreover, this strategy cannot be used for the **mp** measure as the vector weights are not fixed in advance, but optimized for each vector combination. Therefore, novel algorithms are needed that reduce the amount of candidates that need to be considered in a different way.

3.3 Multivariate Correlation Discovery Algorithms

When introducing the tripole correlation measure in 2017, Agrawal and his colleagues also proposed two algorithms for finding strongly correlated tripoles [2]. The first, called CONTRaComplete, prunes the number of candidates that need to be considered by relying on the minimum jump constraint, as defined in Section 2.2. They derive results that relate the jump threshold to a requirement on the strength of the pairwise correlations between vectors; all pairs that do not meet this requirement are pruned from the set of all vector pairs. The remaining vectors are subsequently extended to triplets and evaluated. The second algorithm, called CONTRaFast, extends the first by adding a second pruning step to the set of vector pairs: all pairs $\mathbf{a}, \mathbf{b}$ that are highly similar to another pair $\mathbf{x}, \mathbf{y}$ that is in the set of candidate pairs are removed. $\mathbf{a}, \mathbf{b}$ is considered to be similar to $\mathbf{x}, \mathbf{y}$ when both $\rho(\mathbf{a}, \mathbf{x}) \geq \kappa$ and $\rho(\mathbf{b}, \mathbf{y}) \geq \kappa$. κ can be tuned by the used to trade-off completeness of the result set for computational performance. Due to its enhanced scalability, the authors focus on CONTRaFast for their experiments. CONTRaFast can retrieve 99% of the interesting tripoles within 100 minutes on their experimental dataset containing 10443 Sea Level Pressure time series. Furthermore, the authors analyze the statistical significance of the tripole relationship and demonstrate its relevance to neuroscience and climate science (see also [34] for a study on the application of tripoles to the latter domain).

Although the measure was shown to be very useful, the proposed algorithms suffer from two limitations. First, it is limited to discovering ternary correlations, while the user may want to capture n-ary relationships in the data. Second, the CONTRaFast algorithm comes with no guarantee on the degree of completeness of the result set, and relies on the minimum jump constraint for its pruning power. Although the constraint can aid in finding the most interesting results, ideally the user can decide whether or not the constraint should be applied, and the algorithm for executing the query should not solely depend on constraints for its pruning power.

As a follow-up, Agrawal proposed the multipole measure and two algorithms to find highly correlated multipoles [3]. Both algorithms again rely on the minimum jump constraint, and neither guarantees completeness of the result set. The first, called CoMEt, builds on the observation that candidates can likely be pruned by considering the highest pairwise correlation between (negative counterparts of) the vectors in the set (we refer to [3] for the details). The problem statement is subsequently translated to a maximal clique enumeration problem and a post-processing phase in which weakly-correlated results are removed. The second algorithm, called CoMEtExtended, extends the first by allowing the user to trade-off completeness for computational performance via a parameter ρ_{CE}. The experimental evaluation on 171 Sea Level Pressure time series show that CoMEtExtended is able to recover a large fraction of all interesting multipole relationships within 1.5 hours, and yields much more complete answer sets compared to baseline methods using l_1 regularization and structure learning algorithms. The significance and usefulness of the measure are analyzed via extensive case studies in the neuroscience and climate science domains.

Both algorithms in [3] suffer from similar limitations as those of the tripole paper. First, no guarantees on the completeness of the result set are provided, which can result in interesting relationships being lost, and both algorithms depend on the minimum jump constraint. Ideally, the user should be guaranteed to receive all results satisfying their query, even if no additional constraint is specified. Second, the experiments demonstrate that the scalability of the algorithms is limited to a few hundreds of vectors at most. With the abundance of data available to data scientists these days, the lack of scalability is a strong limitation on the practical value of the algorithm.

Various works exist to find vector sets with a strong Total Correlation. Nguyen et al. [45] propose a measure closely related to Total Correlation, and an accompanying algorithm that can discover groups of columns in a data base that are highly correlated. This algorithm relies on approximating the joint distribution of the vectors with a combination of pairwise joint distributions and the

DensEst density estimation procedure [44]. Subsequently, quasi-cliques are discovered in which most of the pairwise correlations are high. Because of the monotonicity property of entropy, this implies that the Total Correlation value of this group of variables is also high. However, this implication is not an if-and-only-if relationship, meaning that groups in which pairwise Mutual Information values are low, but of which the Total Correlation value is high, are not detected; these are arguably the most interesting cases, as apparently only considering the group of variables *as a whole* results in an interesting relationship. As such, the proposed algorithm is effectively an approximation algorithm that does not guarantee any completeness.

Zhang et al. developed an exact framework for Total Correlation discovery, but this is limited to binary data like gene expression data [60]. Their methods rely on theoretical results to bound the Total Correlation value from (1) the underlying pairwise Mutual Information values, and (2) the Total Correlation of similar vector sets that have already been evaluated. Furthermore, the algorithm relies on the *weak-correlated subset rule*, which imposes that a vector set is only considered interesting if its correlation exceeds τ, *and* the largest correlation of any of its subsets is at most α. As already indicated, such a constraint may not be applicable to all contexts.

Chiang et al. consider the problem of automatically determining a suitable linear multivariate correlation measure for databases with mixed column types [11]. Subsequently, their methodology finds sets of columns that exhibit a strong and statistically significant correlation. In line with this thesis, they specifically focus on finding relatively small sets of correlated vectors (up to 6), as correlations in larger groups of attributes are typically caused by correlations of smaller attribute groups [11, 22]. The relevance of their method is demonstrated by successfully replicating scientific findings published in various fields in a data-driven manner. In another work, Wang et al. [57] propose a multivariate correlation measure defined as $1 - \text{Det}(\mathbf{R})$, where $\text{Det}(\mathbf{R})$ is the determinant of the Pearson correlation matrix $\mathbf{R}$. They apply the measure to multivariate physical law discovery and propose that it can be used to discover new phenomena in a data-driven way. Both of these works stress the potential and relevance of multivariate correlation analysis, however, they do not propose pruning algorithms. As a result, their experiments are conducted on comparatively small data sets (less than 100 vectors in [11], and up to 357 vectors for ternary correlations and dozens of vectors for quaternary correlations in [57]). This research focuses on developing an efficient pruning framework, such that applying multivariate correlation analysis to larger data sets becomes practically feasible.

Another work on Total Correlation by Nguyen addresses Maximal Multivariate Correlation Analysis [46]. The problem statement of this work is different than ours. Instead of finding a diverse result set of highly correlated groups, they address the problem of optimally discretizing real-valued vectors such that the calculated Total Correlation value of *all* variables is maximal. In addition to pairwise correlation analysis, PAA [30] was also utilized for answering multivariate correlation queries for a *fixed* right hand side set of vectors, e.g., queries of the type "get every stock that is highly correlated with (AAPL, IBM, ASML)" [47]. For the multivariate measure that is considered, the method builds upon the observation that a vector with a high multivariate correlation to the fixed RHS must have pairwise correlations with the RHS's vectors that are outside a hyper-ellipsoid that is defined solely by the vectors in the RHS and the correlation threshold τ (see [47] for the details). This area can subsequently be queried using indices like R-trees [21]. However, as our problem considers no fixed right hand side, but instead, the objective is to discover all groups of highly correlated vector sets, the methodology is inapplicable.

3.4 Tangent Research

This study aims to find subsets of features that are highly correlated. The problem of selecting subsets out of a larger feature set is similar to subspace clustering [50] and frequent itemset mining [41]. Notably, Cheng et al. use the notion of interest, which is equivalent to Total Correlation,

as one criterion of a good clustering in their subspace clustering algorithm ENCLUS [10]. As such, their methodologies are worth considering for our problem.

Both the subspace clustering and frequent itemset mining problems are commonly solved using an algorithm inspired by apriori [1], in which the results of a candidate subset can be use to prune supersets of that candidate. Depending on the correlation measure under consideration, and on whether or not additional constraints are provided by the user, the apriori-framework can theoretically be applied to this problem as well. Some correlation measures are monotonic, that is, adding a vector $\mathbf{v}$ to a candidate subset X cannot decrease correlation. Examples are the multipole and Total Correlation measures. Furthermore, if a measure is not monotonic, the irreducibility constraint indicates that the user is not interested in supersets of already correlated subsets. However, apriori-like approaches typically rely on pruning many candidates at the first levels of the search lattice for their effectiveness. As in our case the user is interested in the *strongest* correlated subsets, our algorithm effectively needs to look for outliers, and it is expected that only a small fraction of all candidates satisfies the query. Therefore, only few candidates can be pruned at the lower levels in the search space, and apriori loses much of its effectiveness. Moreover, without the irreducibility constraint, the $\mathbf{mc}(X, Y)$ measure is not suitable for apriori-like techniques as its correlation value is not monotonic in the size of X and Y.

The subset selection problem [14] is also somewhat similar to our problem statement: given a dependent variable $\mathbf{y}$, find a subset X containing k features that minimizes the mean squared error $\min_{\mathbf{w}\in\mathbb{R}^k} ||\mathbf{y} - \mathbf{X}\cdot\mathbf{w}||_2^2$. As the problem is NP-hard in general, heuristics like forward selection, backward elimination or genetic algorithms are commonly used to find a good solution [26, 42]. Alternatively, l_1-regularization is frequently used to find a sparse and interpretable model [54]. Our problem differs from the above in two aspects: first, as we aim to find interesting patterns in the data instead of finding the best predictors for a dependent variable, it is unsupervised rather than supervised. Second, as opposed to finding only the *highest* correlated set of vectors, our goal is to find a *diverse and complete* set of results that exhibit an interesting relation. We stress the importance of the second discrepancy, as finding a diverse answer set allows the domain expert to assess the results on qualitative aspects in order to make decisions or gain insights. l_1-based techniques could in principle be used to find multiple interesting sets by varying the regularization coefficient, but for **mp** it has already been shown that this method only retrieves small fractions of all interestingly correlated vector sets [3]. As such, alternative methods must be developed for out problem statement.

Chapter 4

Correlation Detective

4.1 General Idea and Intuition

As discussed in Section 1.1, the main challenge of the multivariate correlation discovery problem is the vast search space. When considering a query with maximum cardinality $p = l_{\max} + r_{\max}$, there are $\mathcal{O}(\sum_{k=2}^{p} \binom{n}{k})$ vector subsets that need to be considered. Clearly, to efficiently arrive at a solution, the amount of candidates that have to be enumerated needs to be reduced. As such, an efficient algorithm must be able to decide on the status of multiple candidates in a single step.

To this end, we propose the Correlation Detective framework, abbreviated as CD. CD exploits the observation that real data typically contains non-trivial correlations between vectors. For example, closing prices of two stocks that are related to the same corporation are typically very strongly correlated. See Figure 4.1 for an example. Given that the z-normalized closing prices of GOOG and GOOGL are nearly identical, it would make sense to reduce the number of candidates by not considering both time series separately, but only a single representative vector which serves as a proxy for both. Moreover, two stocks that are exposed to similar market risks and opportunities, such as ASML and STMicroelectronics, also show significant correlation. The key question is whether or not these vectors are correlated strongly enough to be represented by a single proxy vector. Whether this is the case depends not only on the correlation between these vectors, but also on the other vectors in X and Y. CD relies on novel theoretical results to dynamically decide whether a *cluster* of vectors can be represented by its centroid without introducing errors, based on the other vectors in X and Y and the correlation between the vectors in the cluster.

The high level workflow of CD is illustrated in Figure 4.2: first, a hierarchical clustering of the vectors is constructed. Then, instead of enumerating over all possible candidates, only combinations of the clusters are enumerated. Subsequently, for each combination of clusters, upper en lower bounds on the correlation values of all candidates that are in the Cartesian product of the clusters are calculated. These bounds depend on the correlations between the cluster centroids and the geometric size of the clusters. CD uses the bounds to decide if the clusters should be added to the result set, can safely be discarded, or if clusters should be split into smaller subclusters to derive tighter bounds.

This chapter contains all of the ingredients needed to build the CD algorithm. In Section 4.2, the initialization phase of the algorithm is discussed, where the data is preprocessed and clustered for the subsequent steps. Then, in Section 4.3, the core Correlation Detective algorithm, used for answering threshold queries, is presented. In Section 4.4, we present the derivation of the formulas for calculating the bounds that are used by CD. Finally, Section 4.5 presents the extensions of the core CD algorithm used for answering top-k and progressive queries and handling optional constraints.

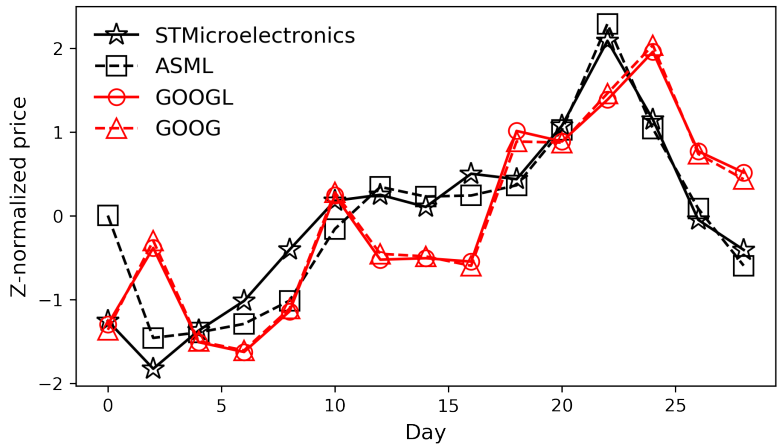


Figure 4.1: Two groups of closely related stocks: ASML and STMicroelectronics are exposed to similar markets, while GOOG and GOOGL participate in the same conglomeration. Figure adapted from Minartz, d'Hondt and Papapetrou, 2021 [39].

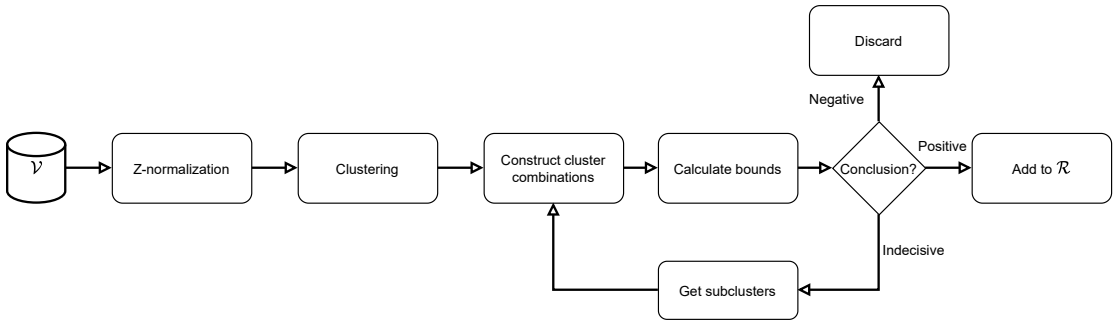


Figure 4.2: Flowchart illustrating the core Correlation Detective Algorithm

To simplify the discussion in the remainder of this chapter, we will refer to the following definitions:

Definition 4.1 (**Cluster Radius**)**.** The radius r of a cluster C is defined as the maximum distance from the cluster's centroid $\mathbf{c}$ to any other point in that cluster:

$$r = \max_{\mathbf{v} \in C} (d(\mathbf{c}, \mathbf{v}))$$

Definition 4.2 (**Cluster Combination**)**.** A cluster combination (abbreviated as CC) is a partitioning of a multiset of clusters $\mathcal{S} = \{C_i\}_{i=1}^{p}$ into two multisets $\mathcal{X} = \{C_i\}_{i=1}^{l_{\max}}$ and $\mathcal{Y} = \{C_i\}_{i=l_{\max}+1}^{p}$ such that $l_{\max}$ and $r_{\max}$ adhere to the correlation pattern provided by the user. For one sided correlation measures, $\mathcal{X} = \mathcal{S}$ and $\mathcal{Y} = \emptyset$. A cluster combination compactly represents the set of candidates that can be formed by the Cartesian product of its clusters.

Definition 4.3 (**Materialization**)**.** A materialization of a cluster combination consists of two sets of vectors that can be formed by picking precisely one vector from each cluster. More specifically, a materialization (X, Y) of a cluster combination $(\mathcal{X}, \mathcal{Y})$ is defined as follows:

$$X = \{\mathbf{v_i} \mid \mathbf{v_i} \in C_i\}_{i=1}^{l_{\max}}$$
$$Y = \{\mathbf{v_i} \mid \mathbf{v_i} \in C_i\}_{i=l_{\max}+1}^{p}$$

As an example of cluster combinations and materializations, consider the clusters $C_1 = \{\mathbf{a}, \mathbf{b}, \mathbf{c}\}$ and $C_2 = \{\mathbf{x}, \mathbf{y}\}$. The cluster combination $\mathcal{X} = \{C_1, C_1, C_2\}$, $\mathcal{Y} = \emptyset$ represents all of its materializations $\{\mathbf{a}, \mathbf{b}, \mathbf{x}\}$, $\{\mathbf{a}, \mathbf{c}, \mathbf{x}\}$, $\{\mathbf{b}, \mathbf{c}, \mathbf{x}\}$, $\{\mathbf{a}, \mathbf{b}, \mathbf{y}\}$, $\{\mathbf{a}, \mathbf{c}, \mathbf{y}\}$ and $\{\mathbf{b}, \mathbf{c}, \mathbf{y}\}$.

4.2 Initialization

Data Preprocessing

As a preprocessing step, CD z-normalizes each vector. That is, every vector $\mathbf{x}$ is transformed to its z-normalized counterpart $\hat{\mathbf{x}}$ as follows:

$$\hat{\mathbf{x}} = \frac{\mathbf{x} - \mu(\mathbf{x})}{\sigma(\mathbf{x})}$$

Z-normalizing all vectors follows the preprocessing steps of earlier works [2, 3] and is a requirement for the mathematical equivalences of Pearson correlation, Euclidean distance and angles between vectors, which are exploited later in this chapter. For the **mp** measure (Eq. 2.6), this preprocessing step is not strictly necessary; subtracting the mean from each vector does not affect the variance of $\mathbf{X} \cdot \mathbf{w}$ and therefore has no effect on the correlation value, and dividing each vector by its standard deviation changes the weights in the weight vector $\mathbf{w}$, but does not change the correlation. However, for the **mc** measure, z-normalization is crucial to arrive at meaningful results. Take as an example two cryptocurrencies, Bitcoin (BTC) and Dogecoin (DOGE). At the time of writing, DOGE trades at \$0.34, while BTC trades at \$47122.90. Assume that the **mc** measure is used to look for financial assets that are highly correlated to $\frac{\text{BTC+DOGE}}{2}$. If both currencies' prices fluctuate by 10% in a given week, the effect of the fluctuations of DOGE would be negligible compared to the much larger fluctuations (in absolute value) of BTC, and the outcomes would essentially be determined by the correlation with BTC only. Z-normalization ensures that all vectors are in the same order of magnitude (in fact, their l_2-norm equals $\sqrt{d}$, d being the dimensionality of the vector). As a result, no signal is lost in the aggregation.

Hierarchical Clustering

Ideally, the clustering approach should meet certain requirements. First, the clustering should be computationally cheap to construct, as to not spend significant computation time on the initialization phase. Second, to derive tight bounds on correlations, the clusters' radii should be as small as possible (this will become apparent in Section 4.4). Third, to be able to prune large numbers of candidates in one step, the clusters should contain many points. Clearly, there is a direct trade off between the second and third desired property of the clusters: adding more vectors to a cluster may increase the cluster's radii.

As a stepping stone towards a hierarchical clustering algorithm, we first discuss a simple non-hierarchical clustering algorithm. To balance the three desired properties of the clustering algorithm, an algorithm inspired by k-means++ [6] is used for each level in the cluster hierarchy. This approach prevents similar time series from being assigned to different clusters as much as possible: for each vector $\mathbf{v}$ and parameter ϵ, if a cluster C with $d(\mathbf{v}, C.\text{centroid}) \leq \epsilon$ already exists, then $\mathbf{v}$ is assigned to C. If this is not the case, and if the number of clusters is smaller than a parameter K, a new cluster is created with $\mathbf{v}$ as its centroid. If the number of clusters is equal to K, and no cluster C was found for which $d(\mathbf{v}, C.\text{centroid}) \leq \epsilon$, then $\mathbf{v}$ is simply assigned to the cluster of which the centroid is closest to $\mathbf{v}$. Note that in contrast to the standard K-means algorithm, we perform only one iteration of setting the centroids, as experiments have shown that performing more iterations had no effect on the performance of CD.

Algorithm 1: ClusteringOneLevel(V, ϵ, K)
Input: a set of d-dimensional vectors V, a distance threshold ϵ, the maximum number of clusters $K > 1$.
Output: A set of clusters $\mathcal{S}$, such that $|\mathcal{S}| \leq K$.

```
1  S ← ∅
2  for v ∈ V do
3  |  d* ← ∞
4  |  for C ∈ S do                                  // Find closest existing cluster
5  |  |  dist ← d(v, C.centroid)
6  |  |  if dist < dist* then
7  |  |  |  C* ← C
8  |  |  |  dist* ← dist
9  |  if (dist* > ε) ∧ (|S| < K) then               // Initialize a new cluster
10 |  |  Initialize a new cluster C_new with v as centroid
11 |  |  S ← S ∪ {C_new}
12 |  else                                          // Assign to closest cluster
13 |  |  C* ← C* ∪ {v}
14 return S
```

Now, we extend the above method to construct a hierarchical clustering. The clustering hierarchy is initialized as one root cluster containing all vectors. Then, in a recursive step, the data points contained in the cluster are reclustered into smaller clusters. To this end, the parameter ϵ is multiplied by some multiplier α, $0 < \alpha < 1$, forcing Algorithm 1 to make smaller clusters. In order to prevent a coincidentally poor clustering having a detrimental effect on CD's performance, Algorithm 1 is executed n_rep times, and the clustering with the lowest sum of squared distances from each vector to its cluster's centroid is kept as the final clustering. The recursion stops when a cluster contains only one vector, or if the depth of a cluster in the hierarchy equals a threshold $h_{max} - 1$, after which each vector in the cluster is assigned to a subcluster containing only that vector. The pseudocode is given in Algorithm 2.

Complexity

The complexity of the algorithm's initialization phase is negligible compared to the complexity of the problem as a whole. Z-normalization of all vectors requires iterating over d dimensions for all n vectors, resulting in a complexity of $\mathcal{O}(nd)$. The complexity of the hierarchical clustering algorithm is also linear in the number of vectors n. Observe that at each level in the clustering tree, each vector is compared to at most $k \cdot \text{n\_rep}$ cluster centroids; as such, for any given level in the clustering tree with N clusters in total, the combined complexity of all calls of Algorithm 1 from lines 14 and 16 of Algorithm 2 for that level is $\mathcal{O}\left(\sum_{i=1}^{N} |C_i| \cdot k \cdot \text{n\_rep} \cdot d\right) = \mathcal{O}(n \cdot k \cdot d \cdot \text{n\_rep})$. Since the clustering tree contains at most $h_{\max}$ levels, the overall complexity of Algorithm 2 is $\mathcal{O}(n \cdot k \cdot d \cdot \text{n\_rep} \cdot h_{\max})$. Practical experiments confirm that the initialization phase takes only a fraction of the total running time of our algorithm.

Algorithm 2: HierarchicalClustering(P, ϵ, K, α, h_{max}, n_rep)
Input: a parent cluster P, a distance threshold ϵ, a distance threshold multiplier α, the maximum number of subclusters per level K, the maximum height of the clustering tree h_{max}, number of clustering repetitions n_rep.

```
1  score* ← ∞
2  S* ← ∅
3  for i=1 to n_rep do                    // Perform repetitions and find best clustering
4  |  Randomly permute the vectors in P
5  |  S ← ClusteringOneLevel(P, ε, K)
6  |  score ← Σ_{C∈S} Σ_{v∈C} d²(v, C.centroid)
7  |  if score < score* then
8  |  |  S* ← S
9  |  |  score* ← score
10 ε ← α · P.radius                       // Make smaller subclusters
11 for C ∈ S* do
12 |  if |C| > 1 then
13 |  |  if P.depth < h_max − 1 then
14 |  |  |  HierarchicalClustering(C, ε, K, α, h_max)
15 |  |  else                            // generate one subcluster for each vector
16 |  |  |  HierarchicalClustering(C, 0, |C|, α, h_max)
```

4.3 Correlation Detective for Threshold Queries

In this section, we discuss the core of the Correlation Detective algorithm for answering threshold queries. The input to the CD algorithm consists of the z-normalized data, the hierarchical clustering produced by Algorithm 2, the user-defined correlation pattern and a correlation threshold τ. CD then proceeds to construct cluster combinations of increasing cardinalities, consisting of only root clusters, until the maximum cardinality as specified by the correlation pattern is reached. For example, if the correlation pattern is $\mathbf{mc}(1,3)$, CD will consider $(\{C_{\text{root}}\}, \{C_{\text{root}}\})$, $(\{C_{\text{root}}\}, \{C_{\text{root}}, C_{\text{root}}\})$ and $(\{C_{\text{root}}\}, \{C_{\text{root}}, C_{\text{root}}, C_{\text{root}}\})$ in order (as illustrated in Figure 4.3).

For each cluster combination, CD computes lower and upper bounds LB and UB on *any* materialization of that cluster combination (line 1 of Algorithm 3) in one computational step; the derivation of the formulas to calculate these bounds will be presented in Section 4.4. If $LB \geq \tau$, the CC is *decisive positive*, meaning that any materialization of the CC is guaranteed to satisfy the user query, and as such, all materializations are added to the result set $\mathcal{R}$ (line 3). Analogously, if $UB < \tau$, the CC is *decisive negative*, guaranteeing that all materializations do not satisfy the query. As such, the CC can safely be discarded, and all candidates that are contained in the CC

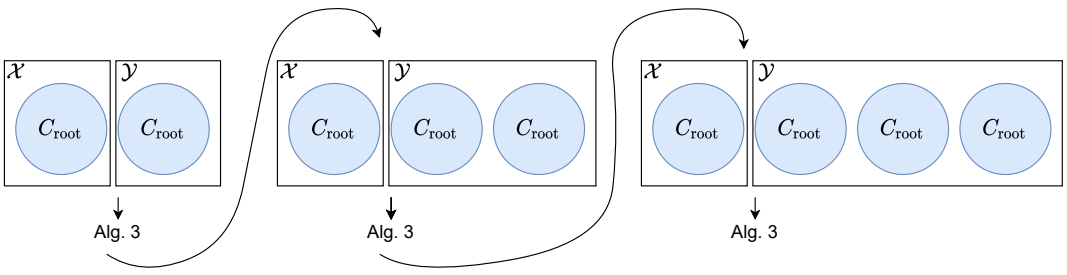
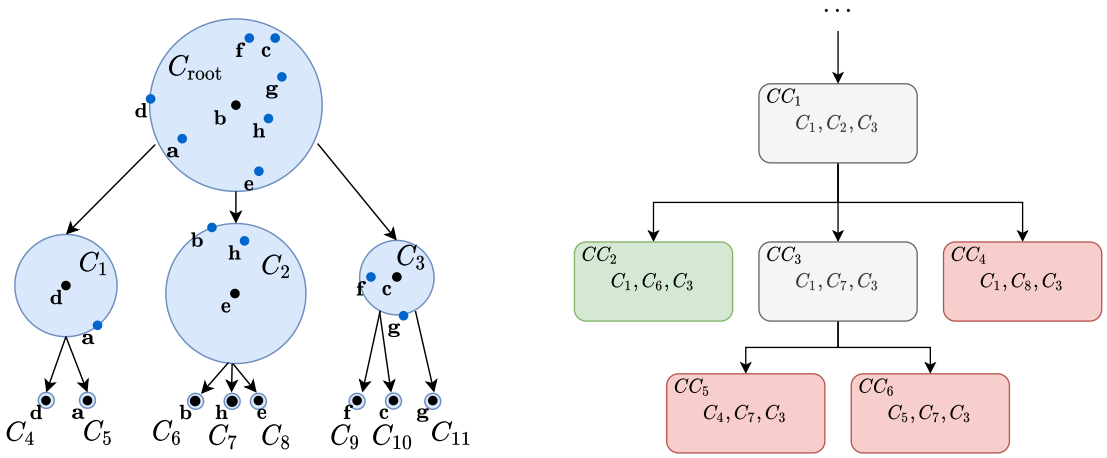


Figure 4.3: Conceptual illustration of CD processing correlation patterns in order of increasing complexity. For each pattern, CD starts by considering the cluster combination containing only root clusters.

are pruned (line 5). Finally, if $LB < \tau \leq UB$, the CC is *indecisive*. In this case, CD identifies the cluster $C_{\max}$ that has the largest radius, gets its subclusters, and for each of those recursively evaluates a new cluster combination in which $C_{\max}$ is replaced by the subcluster (line 7-11). In case two or more clusters in the CC have exactly the same radius (for example because they are the same clusters), $C_{\max}$ is chosen at random from those clusters.

Algorithm 3: THRESHOLDQUERY($\mathcal{X}$, $\mathcal{Y}$, $Corr$, τ)
Input: Multisets of clusters $\mathcal{X}$ and $\mathcal{Y}$, correlation measure $Corr$, correlation threshold τ.
1 $(LB, UB) \leftarrow$ CALCBOUNDS$(\mathcal{X}, \mathcal{Y}, Corr)$ // As in Section 4.4
2 **if** $LB \geq \tau$ **then**
3 | Add $(\mathcal{X}, \mathcal{Y})$ to the result set
4 **else if** $UB < \tau$ **then**
5 | Discard $(\mathcal{X}, \mathcal{Y})$
6 **else**
| // Replace largest cluster with subclusters and recurse
7 | $C_{max} \leftarrow \arg\max_{C \in \mathcal{X} \cup \mathcal{Y}} \{C.radius\}$
8 | Set $SC \leftarrow C_{max}.subclusters$
9 | **for** $S \in SC$ **do**
10 | | $(\mathcal{S}'_l, \mathcal{S}'_r) \leftarrow (\mathcal{X}, \mathcal{Y})$ with C_{max} replaced by S
11 | | THRESHOLDQUERY$((\mathcal{S}'_l, \mathcal{S}'_r), Corr, \tau)$



(a) Clustering tree constructed by Algorithm 2. (b) Pruning mechanism of Algorithm 3.

Figure 4.4: Toy example illustrating the Correlation Detective Algorithm.

We illustrate the functioning of our algorithm using Figure 4.4. Assume Algorithm 3 has just been called on cluster combination CC_1 of Figure 4.4b, containing C_1, C_2 and C_3, and that $\tau = 0.8$. CD calculates the bounds (line 1), which turn out to be $[0.6, 0.95]$, and as such the CC is indecisive. CD will now retrieve the subclusters of C_2, which has the largest radius of the three clusters, and generates new CCs by replacing C_2 with its subclusters, and recursively calls Algorithm 3 on the new CCs (lines 7-11). The bounds on CC_2 turn out to be $[0.85, 0.9]$, and as such, the CC is added to the result set (line 3). The bounds on CC_4 are $[0.65, 0.75]$, and therefore the CC can safely be discarded (line 5). Finally, the bounds on CC_3 are $[0.7, 0.82]$, and CD recurses by splitting C_1 in subclusters. The new candidates CC_5 and CC_6 have bounds $[0.75, 0.78]$ and $[0.72, 0.74]$ respectively, and as such both CCs are decisive negative.

Note that, even in this toy example where each cluster contains only few vectors, CD was able to evaluate $|C_1 \times C_2 \times C_3| = 18$ candidate subsets by considering only 6 cluster combinations. Decisive cluster combinations are typically found at high levels in the clustering hierarchy, allowing CD to prune many candidates in few comparisons.

Implementation details. Both the **mc** and **mp** measures are in some sense symmetric: $\mathbf{mc}(\{\mathbf{x}\}, \{\mathbf{y}, \mathbf{z}\})$ is the same as $\mathbf{mc}(\{\mathbf{x}\}, \{\mathbf{z}, \mathbf{y}\})$, and $\mathbf{mp}(\{\mathbf{x}, \mathbf{y}, \mathbf{z}\})$ is the same as $\mathbf{mp}(\{\mathbf{y}, \mathbf{x}, \mathbf{z}\})$, and so on. Our implementation exploits this symmetry by skipping redundant cluster combinations. If $\mathcal{X} = \{C_1, C_2, C_3\}$ is already considered, then $\mathcal{X}' = \{C_2, C_1, C_3\}$ can safely be discarded, as all materializations of $\mathcal{X}'$ are also materializations of $\mathcal{X}$.

4.4 Calculating Bounds

In this section, we elaborate on two different methods to derive the bounds of line 1 in Algorithm 3. The first approach (which we refer to as *theoretical bounds*) has constant complexity in the size of the clusters, whereas the complexity of the second approach (referred to as *empirical bounds*) is quadratic in the amount of vectors in the clusters, but takes into account more information about the geometrical shape of the clusters to derive tighter bounds.

The discussion in this section is structured as follows: first, we present a lemma to bound the pairwise correlations between two clusters using the theoretical bounds. Then, we extend the discussion to multivariate correlations, and show the relevance of the lemma in deriving bounds on **mc** and **mp** correlations. Finally, we present the empirical bounds, as well as an intuition on why these are beneficial for CD's efficiency when dealing with high-dimensional data.

Theoretical Bounds

We will see that bounds on the pairwise correlation between any materialization of two clusters can be used to find bounds on **mc** and **mp** correlation. Therefore, we first formulate the following lemma:

Lemma 4.1 (**Pairwise Theoretical Bounds**)**.** Let C_1, C_2 be two clusters, with centroids $\mathbf{c_1}$ and $\mathbf{c_2}$ and radii r_1 and r_2, and let $\theta_{\mathbf{c_1},\mathbf{c_2}} = \arccos(\rho(\mathbf{c_1}, \mathbf{c_2}))$. Let $\mathbf{v_1}, \mathbf{v_2}$ be any materialization of the two clusters. Then, the Pearson correlation $\rho(\mathbf{v_1}, \mathbf{v_2})$ can be bounded as follows:

$$\cos\left(\theta^{max}_{\mathbf{v_1},\mathbf{v_2}}\right) \leq \rho(\mathbf{v_1}, \mathbf{v_2}) \leq \cos\left(\theta^{min}_{\mathbf{v_1},\mathbf{v_2}}\right)$$

where

$$\theta^{max}_{\mathbf{v_1},\mathbf{v_2}} = \min\left(\pi, \theta_{\mathbf{c_1},\mathbf{c_2}} + \theta_1 + \theta_2\right)$$

$$\theta^{min}_{\mathbf{v_1},\mathbf{v_2}} = \max\left(0, \theta_{\mathbf{c_1},\mathbf{c_2}} - \theta_1 - \theta_2\right)$$

$$\theta_i = \arccos\left(1 - \frac{r_i^2}{2d}\right)$$

.

Proof. Recall from Section 2.1 that the Pearson correlation $\rho(\mathbf{x}, \mathbf{y})$ between two z-normalized vectors $\mathbf{x}$ and $\mathbf{y}$ is a function of the Euclidean distance between those vectors: $\rho(\mathbf{x}, \mathbf{y}) = 1 - \frac{1}{2d} d^2(\mathbf{x}, \mathbf{y})$. Now, since the radius r of a cluster is defined as the largest distance from any vector in the cluster to the centroid, $1 - \frac{r^2}{2d}$ corresponds to the lowest Pearson correlation between any vector in the cluster and the centroid. Since arccos is a monotonically decreasing function, θ_i is the largest

angle of any vector in C_i with the cluster centroid $\mathbf{c_i}$.

Now, from the (reverse) triangle inequality, it is straightforward to see that the angle $\theta_{\mathbf{v_1},\mathbf{v_2}}$ between $\mathbf{v_1}$ and $\mathbf{v_2}$ can be bounded:

$$\begin{aligned}\theta_{\mathbf{c_1},\mathbf{c_2}} - \theta_1 - \theta_2 &\leq \theta_{\mathbf{c_1},\mathbf{c_2}} - \theta_{\mathbf{v_1},\mathbf{c_1}} - \theta_{\mathbf{v_2},\mathbf{c_2}} \\ &\leq \theta_{\mathbf{v_1},\mathbf{v_2}} \\ &\leq \theta_{\mathbf{c_1},\mathbf{c_2}} + \theta_{\mathbf{v_1},\mathbf{c_1}} + \theta_{\mathbf{v_2},\mathbf{c_2}} \\ &\leq \theta_{\mathbf{v_1},\mathbf{v_2}} + \theta_1 + \theta_2\end{aligned}$$

Furthermore, under the convention that all angles between two vectors are in $[0, \pi]$, we can rewrite this as

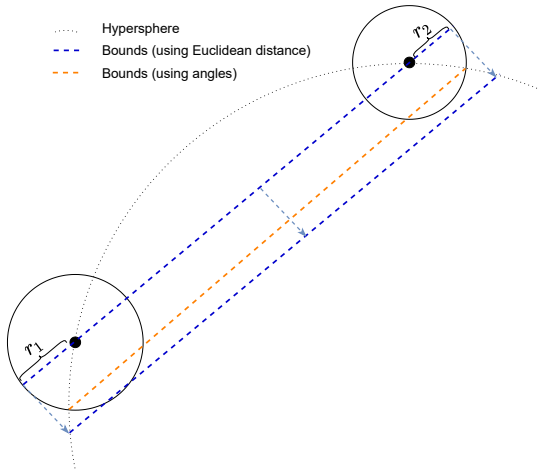
$$\max(0, \theta_{\mathbf{c_1},\mathbf{c_2}} - \theta_1 - \theta_2) \leq \theta_{\mathbf{v_1},\mathbf{v_2}} \leq \min(\pi, \theta_{\mathbf{c_1},\mathbf{c_2}} + \theta_1 + \theta_2)$$

Since cos is a monotonically decreasing function on $[0, \pi]$, and $\cos(\theta_{\mathbf{v_1},\mathbf{v_2}}) = \rho(\mathbf{v_1}, \mathbf{v_2})$, we get the final result:

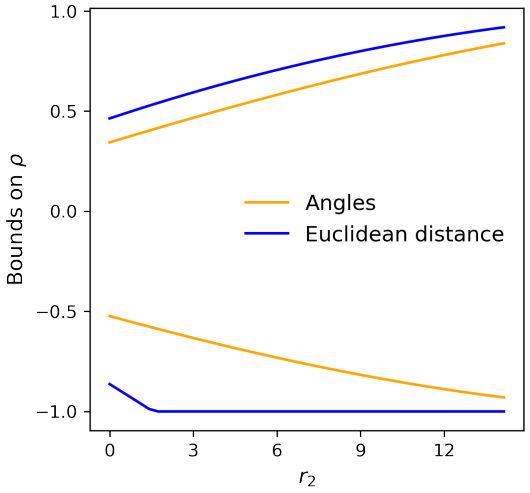
$$\cos(\min(\pi, \theta_{\mathbf{c_1},\mathbf{c_2}} + \theta_1 + \theta_2)) \leq \rho(\mathbf{v_1}, \mathbf{v_2}) \leq \cos(\max(0, \theta_{\mathbf{c_1},\mathbf{c_2}} - \theta_1 - \theta_2))$$

□

The above lemma essentially boils down to a similar idea as in Mueen et al. to bound correlations using the triangle inequality [43]. However, we exploit the fact that all vectors are z-normalized by observing that all vectors have the same l_2-norm and thus lie on a hypersphere. Therefore, the triangle inequality can be applied to angles instead of Euclidean distance, resulting in slightly tighter bounds. An intuition for this is depicted in Figure 4.5. Figure 4.5a illustrates that the largest possible Euclidean distance between two clusters is smaller when taking into account that all vectors are located on a hypersphere; as a result, the lower bound on correlation is higher when derived using angles. Figure 4.5b gives an intuition on the difference in tightness of bounds derived by using both angles and Euclidean distance.



(a) Schematic illustration in $\mathbb{R}^2$ of the lower bound on correlation derived by applying the triangle inequality to both Euclidean distance and angles.



(b) Lower and upper bounds derived by applying the triangle inequality to both Euclidean distance and angles, for varying radius r_2 ($d = 500$, $r_1 = 10$, $\rho(\mathbf{c_1}, \mathbf{c_2}) = -0.1$).

Figure 4.5: Applying the triangle inequality to angles (orange) rather than Euclidean distance (blue) yields tighter bounds.

Lemma 4.1 allows us to bound pairwise Pearson correlations between clusters. For instance, in the example of Figure 4.4a, the correlation between **a** and **h** can be bounded if we know the angle between **d** and **e** and the radii of C_1 and C_2 which can easily be cached during the initialization phase. If these bounds are sufficiently tight, it can be determined whether $\mathbf{a}, \mathbf{h}$ should be put in the result set without actually calculating the correlation between the two vectors. If the bounds are indecisive, splitting the clusters into subclusters generally decreases the clusters' radii, thereby tightening the bounds.

Bounding the mc Measure

We will now elaborate on how Lemma 4.1 can be used to bound **mc** correlations.

Theorem 4.1 (Bounds on mc). Consider a cluster combination $(\mathcal{X}, \mathcal{Y})$. For any pair of clusters $C_i, C_j \in \mathcal{X} \cup \mathcal{Y}$, let $l(C_i, C_j)$ and $u(C_i, C_j)$ denote valid upper an lower bounds on any materialization of the two clusters, i.e. $l(C_i, C_j) \leq \min_{\mathbf{x} \in C_i, \mathbf{y} \in C_j} \rho(\mathbf{x}, \mathbf{y})$ and $u(C_i, C_j) \geq \max_{\mathbf{x} \in C_i, \mathbf{y} \in C_j} \rho(\mathbf{x}, \mathbf{y})$. Note that these can be derived via Lemma 4.1. Furthermore, let $L(\mathcal{S}_1, \mathcal{S}_2) = \sum_{C_i \in \mathcal{S}_1, C_j \in \mathcal{S}_2} l(C_i, C_j)$ and let $U(\mathcal{S}_1, \mathcal{S}_2) = \sum_{C_i \in \mathcal{S}_1, C_j \in \mathcal{S}_2} u(C_i, C_j)$. Then, any materialization (X, Y) of the cluster combination $(\mathcal{X}, \mathcal{Y})$ can be bounded as follows:

1. if $L(\mathcal{X}, \mathcal{Y}) \geq 0$:

$$\frac{L(\mathcal{X}, \mathcal{Y})}{\sqrt{U(\mathcal{X}, \mathcal{X})}\sqrt{U(\mathcal{Y}, \mathcal{Y})}} \leq \mathbf{mc}(X, Y) \leq \frac{U(\mathcal{X}, \mathcal{Y})}{\sqrt{\max(\epsilon, L(\mathcal{X}, \mathcal{X}))}\sqrt{\max(\epsilon, L(\mathcal{Y}, \mathcal{Y}))}}$$

2. if $U(\mathcal{X}, \mathcal{Y}) \leq 0$:

$$\frac{L(\mathcal{X}, \mathcal{Y})}{\sqrt{\max(\epsilon, L(\mathcal{X}, \mathcal{X}))}\sqrt{\max(\epsilon, L(\mathcal{Y}, \mathcal{Y}))}} \leq \mathbf{mc}(X, Y) \leq \frac{U(\mathcal{X}, \mathcal{Y})}{\sqrt{U(\mathcal{X}, \mathcal{X})}\sqrt{U(\mathcal{Y}, \mathcal{Y})}}$$

3. else:

$$\frac{L(\mathcal{X}, \mathcal{Y})}{\sqrt{\max(\epsilon, L(\mathcal{X}, \mathcal{X}))}\sqrt{\max(\epsilon, L(\mathcal{Y}, \mathcal{Y}))}} \leq \mathbf{mc}(X, Y) \leq \frac{U(\mathcal{X}, \mathcal{Y})}{\sqrt{\max(\epsilon, L(\mathcal{X}, \mathcal{X}))}\sqrt{\max(\epsilon, L(\mathcal{Y}, \mathcal{Y}))}}$$

where $\epsilon > 0$ is a sufficiently small scalar.

Proof. The **mc** correlation function can be rewritten as follows:

$$\begin{aligned}
\mathbf{mc}(X,Y) &= \rho(\mathbf{Avg}(\mathbf{X}), \mathbf{Avg}(\mathbf{Y})) \\
&= \frac{\sum_{i=1}^{d} Avg(X)_i \cdot Avg(Y)_i}{d \cdot \sigma(\mathbf{Avg}(\mathbf{X})) \cdot \sigma(\mathbf{Avg}(\mathbf{Y}))} \\
&= \frac{\sum_{i=1}^{d} \left(\sum_{\mathbf{x}\in X} x_i\right) \cdot \left(\sum_{\mathbf{y}\in Y} y_i\right)}{d \cdot \sigma\left(\sum_{\mathbf{x}\in X} \mathbf{x}\right) \cdot \sigma\left(\sum_{\mathbf{y}\in Y} \mathbf{y}\right)} \\
&= \frac{\sum_{\mathbf{x}\in X} \sum_{\mathbf{y}\in Y} \frac{1}{d} \sum_{i=1}^{d} x_i \cdot y_i}{\sqrt{\sum_{\mathbf{i},\mathbf{j}\in X} \mathrm{Cov}(\mathbf{i},\mathbf{j})} \cdot \sqrt{\epsilon, \sum_{\mathbf{i},\mathbf{j}\in Y} \mathrm{Cov}(\mathbf{i},\mathbf{j})}} \\
&= \frac{\sum_{\mathbf{x}\in X} \sum_{\mathbf{y}\in Y} \rho(\mathbf{x},\mathbf{y})}{\sqrt{\sum_{\mathbf{i},\mathbf{j}\in X} \rho(\mathbf{i},\mathbf{j})} \cdot \sqrt{\sum_{\mathbf{i},\mathbf{j}\in Y} \rho(\mathbf{i},\mathbf{j})}}
\end{aligned}$$

Since $\mathcal{S}$ is a set of clusters rather than fixed vectors, the correlations $\rho(\cdot,\cdot)$ are not fixed. However, these can be lower- and upper-bounded with $l(\cdot,\cdot)$ and $u(\cdot,\cdot)$ respectively. Then, the numerator is bounded as follows:

$$L(\mathcal{X},\mathcal{Y}) \leq \sum_{\mathbf{x}\in X, \mathbf{y}\in Y} \rho(\mathbf{x},\mathbf{y}) \leq U(\mathcal{X},\mathcal{Y})$$

Similarly, the denominator could bounded as follows:

$$\sqrt{L(\mathcal{X},\mathcal{X})}\sqrt{L(\mathcal{Y},\mathcal{Y})} \leq \sqrt{\sum_{\mathbf{i},\mathbf{j}\in X} \rho(\mathbf{i},\mathbf{j})} \cdot \sqrt{\sum_{\mathbf{i},\mathbf{j}\in Y} \rho(\mathbf{i},\mathbf{j})} \leq \sqrt{U(\mathcal{X},\mathcal{X})}\sqrt{U(\mathcal{Y},\mathcal{Y})}$$

Notice that $L(\mathcal{X},\mathcal{X})$ might be less than or equal to zero. We know that, for any candidate sets X and Y for which $\mathbf{mc}(X,Y)$ is properly defined, $\sigma(\sum_{\mathbf{x}\in X} \mathbf{x}) > 0$ and $\sigma(\sum_{\mathbf{y}\in Y} \mathbf{y}) > 0$ holds. Consequently, for sufficiently small $\epsilon > 0$, it holds that $\sum_{\mathbf{i},\mathbf{j}\in X} \rho(\mathbf{i},\mathbf{j}) \geq \epsilon$ and $\sum_{\mathbf{i},\mathbf{j}\in Y} \rho(\mathbf{i},\mathbf{j}) \geq \epsilon$. Therefore, the denominator can be bounded as:

$$\sqrt{\max\left(\epsilon, L(\mathcal{X},\mathcal{X})\right)}\sqrt{\max\left(\epsilon, L(\mathcal{Y},\mathcal{Y})\right)} \leq \sqrt{\sum_{\mathbf{i},\mathbf{j}\in X} \rho(\mathbf{i},\mathbf{j})} \cdot \sqrt{\sum_{\mathbf{i},\mathbf{j}\in Y} \rho(\mathbf{i},\mathbf{j})} \leq \sqrt{U(\mathcal{X},\mathcal{X})}\sqrt{U(\mathcal{Y},\mathcal{Y})}$$

The final result follows by distinguishing the three cases as presented in the theorem. □

Implementation details. Theorem 4.1 can be used by CD to bound all materializations of a cluster combination by calculating the pairwise correlations between the clusters' centroids and deriving lower and upper pairwise bounds via Lemma 4.1. In our implementation, when calculating $L(\mathcal{X},\mathcal{X})$ and $L(\mathcal{Y},\mathcal{Y})$, we substitute $l(C_i,C_j)$ with 1 if $i = j$, as the correlation of any vector picked from the i'th cluster with itself is 1, and thus for any materialization, $l(C_i,C_i) = 1$ is a valid lower bound on this pairwise correlation. We stress that this is not the case when $\mathcal{X}$ contains the same cluster multiple times and $i \neq j$: in this case, a materialization may consist of

different vectors picked from the i'th and j'th clusters, even though these clusters contain exactly the same vectors. In this case, $l(C_i, C_j)$ is calculated using Lemma 4.1 as for any other cluster pair.

Moreover, if a cluster combination is indecisive, and a cluster is split into its subclusters, the pairwise bounds for every pair of clusters not containing that cluster remain the same. To avoid redundant computational effort, these are cached, and can be re-used in later computations. As the radii of the subclusters are in general smaller than their parent's radius, the bounds on the pairwise correlations involving the subcluster will typically become tighter, thereby also tightening the bounds on $\mathbf{mc}(X, Y)$.

Bounding the mp Measure

Similarly, Lemma 4.1 also serves as a stepping stone for bounding **mp** correlations. We derive the following result:

Theorem 4.2 (**Bounds on mp**). Consider a cluster combination $(\mathcal{X}, \mathcal{Y})$ where $\mathcal{Y} = \emptyset$. For any pair of clusters $C_i, C_j \in \mathcal{X}$, let $l(C_i, C_j)$ and $u(C_i, C_j)$ denote valid bounds on the pairwise correlations of any materialization of the two clusters, i.e. $l(C_i, C_j) \leq \min_{\mathbf{x} \in C_i, \mathbf{y} \in C_j} \rho(\mathbf{x}, \mathbf{y})$ and $u(C_i, C_j) \geq \max_{\mathbf{x} \in C_i, \mathbf{y} \in C_j} \rho(\mathbf{x}, \mathbf{y})$. Note that these can be derived via Lemma 4.1. Furthermore, let the symmetric matrices $\mathbf{L}$ and $\mathbf{U}$ be such that all their elements $l_{ij} = l(C_i, C_j)$ and $u_{ij} = u(C_i, C_j)$. Then, any materialization X of $\mathcal{X}$ can be bounded as follows:

$$1 - \lambda_{min}\left(\frac{\mathbf{L}+\mathbf{U}}{2}\right) - \frac{1}{2}\left\|\mathbf{U}-\mathbf{L}\right\|_2 \leq \mathbf{mp}(X) \leq 1 - \lambda_{min}\left(\frac{\mathbf{L}+\mathbf{U}}{2}\right) + \frac{1}{2}\left\|\mathbf{U}-\mathbf{L}\right\|_2$$

where $\lambda_{min}(\mathbf{A})$ denotes the smallest eigenvalue of a matrix $\mathbf{A}$.

Proof. The multipole correlation $\mathbf{mp}(X)$ can be rewritten as [3]:

$$\mathbf{mp}(X) = 1 - \lambda_{min}(\mathbf{R})$$

where $\lambda_{min}(\mathbf{A})$ denotes the smallest eigenvalue of a matrix $\mathbf{A}$, and $\mathbf{R}$ is the correlation matrix of the vectors in X. Note that, as $\mathcal{X}$ is a set of clusters, the elements of $\mathbf{R}$, r_{ij}, are not fixed. In what follows, let $\mathbf{R}$ denote the correlation matrix of any materialization X of $\mathcal{X}$. Furthermore, for conciseness, let $\mathbf{M} = \frac{\mathbf{U}+\mathbf{L}}{2}$ and $\mathbf{E} = \mathbf{R} - \mathbf{M}$. Then:

$$\lambda_{min}(\mathbf{R}) = \lambda_{min}(\mathbf{M} + \mathbf{E})$$

Using Weyl's inequality [32], we can derive:

$$|\lambda_{min}(\mathbf{M}+\mathbf{E}) - \lambda_{min}(\mathbf{M})| \leq \|\mathbf{E}\|_2 \Leftrightarrow |\lambda_{min}(\mathbf{R}) - \lambda_{min}(\mathbf{M})| \leq \|\mathbf{E}\|_2 \tag{4.1}$$

To complete the proof, it remains to be shown that $\|\mathbf{E}\|_2 \leq \frac{1}{2}\|\mathbf{U}-\mathbf{L}\|_2$. To this end, let $\mathbf{x}^* = \arg\max_{\|\mathbf{x}\|_2=1} \|\mathbf{E}\mathbf{x}\|_2$, and let $\mathbf{y}$ denote the vector containing the absolute values of the elements in $\mathbf{x}^*$: $y_i = |x_i^*|$. Note that also $\|\mathbf{y}\|_2 = 1$. Then:

$$
\begin{aligned}
\|\mathbf{E}\|_2 &= \max_{\|\mathbf{x}\|_2=1} \|\mathbf{E}\mathbf{x}\|_2 \\
&= \|\mathbf{E}\mathbf{x}^*\|_2 \\
&= \sqrt{\sum_{i=1}^{p} \left(\sum_{j=1}^{p} \left(r_{ij} - \frac{u_{ij} + l_{ij}}{2} \right) x_j^* \right)^2} \\
&\leq \sqrt{\sum_{i=1}^{p} \left(\sum_{j=1}^{p} \left| r_{ij} - \frac{u_{ij} + l_{ij}}{2} \right| \left| x_j^* \right| \right)^2} \\
&\leq \sqrt{\sum_{i=1}^{p} \left(\sum_{j=1}^{p} \left| u_{ij} - \frac{u_{ij} + l_{ij}}{2} \right| y_j \right)^2} \\
&= \sqrt{\sum_{i=1}^{p} \left(\sum_{j=1}^{p} \left(\frac{u_{ij} - l_{ij}}{2} \right) y_j \right)^2} \\
&= \left\| \left(\frac{\mathbf{U} - \mathbf{L}}{2} \right) \mathbf{y} \right\|_2 \\
&\leq \max_{\|\mathbf{w}\|_2=1} \left\| \left(\frac{\mathbf{U} - \mathbf{L}}{2} \right) \mathbf{w} \right\|_2 \\
&= \frac{1}{2} \|\mathbf{U} - \mathbf{L}\|_2
\end{aligned}
$$

Substituting this result in Equation 4.1 completes the proof.

□

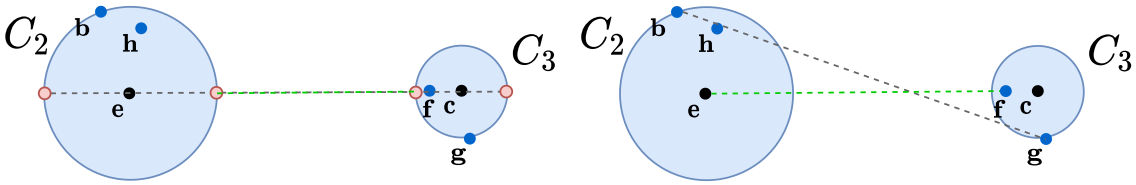
Implementation details. Similar to the calculation of the **mc** bounds, we substitute $l(C_i, C_j)$ with 1 if $i = j$, as this is a valid lower bound on the correlation of a vector with itself for any materialization. Furthermore, progressively increasing the tightness of pairwise bounds by splitting clusters into their subclusters reduces $\|\mathbf{U} - \mathbf{L}\|_2$, which tightens the multipole bound.

Empirical Bounds

Lemma 4.1 equips CD with an elegant and simple method to compute the pairwise bounds $l(C_i, C_j)$ and $u(C_i, C_j)$, regardless of the number of vectors in the clusters. However, especially in high-dimensional data, these bounds are very pessimistic. Consequently, the bounds on the multivariate correlations tend to be loose as well. This is because the bounds have to take into account the worst possible case, where vectors are located precisely on the C_i's edge such that they are as close and as far as possible from C_j, and vice versa. The closest possible pair of vectors corresponds to $u(C_i, C_j)$, whereas the the furthest possible pair corresponds to $l(C_i, C_j)$, as illustrated in Figure 4.6a with red circles. Observe that the hypothetical worst-case is much worse than the real highest and lowest correlations. In higher-dimensional space, the discrepancy typically increases.

To see why this is the case, we can examine a geometrical perspective. Consider vectors in $\mathbb{R}^d$. As all vectors are z-normalized, they have identical l_2 norms, and as such are embedded on a hypersphere $\mathbb{S}^{d-1}$. In order to reach the theoretical upper bound on ρ, there would need to be vectors in both clusters that are exactly at the clusters' edges, but also in each of the $d-1$ dimensions as close as possible to the other cluster (corresponding to the red points in Figure 4.6a). For large

d, this becomes increasingly unlikely. The same intuition holds for the lower bound.



(a) Theoretical bounds.

(b) Empirical bounds.

Figure 4.6: Intuitive illustration of the difference between the theoretical and empirical upper (green) and lower (grey) bounds.

The *empirical bounds* approach builds upon the observation that in real data, the correlations between all materializations of two clusters are often strongly concentrated around the correlation of the cluster centroids. In order to still guarantee the correctness and completeness of CD, the approach works as follows: first, an extra step is added to the initialization phase, where all pairwise correlations between vectors are calculated. Note that some of these have already been cached during the clustering phase and do not need to be recomputed. The pairwise correlations are then stored in an upper-triangular matrix for later use. Then, during the execution of Algorithm 3, instead of using Lemma 4.1 to calculate the pairwise bounds between clusters, $l(C_i, C_j)$ and $u(C_i, C_j)$ are calculated by simply enumerating over all materializations of the two clusters: $l(C_i, C_j) = \min_{\mathbf{x} \in C_i, \mathbf{y} \in C_j} \rho(\mathbf{x}, \mathbf{y})$ and $u(C_i, C_j) = \max_{\mathbf{x} \in C_i, \mathbf{y} \in C_j} \rho(\mathbf{x}, \mathbf{y})$, where $\rho(\cdot, \cdot)$ is retrieved from the matrix storing the pairwise correlations. The resulting bounds are illustrated in Figure 4.6b. Note that u and l are trivially valid upper and lower bounds on any materialization of C_i, C_j, and as such can be used as inputs to Theorems 4.1 and 4.2. Moreover, observe that the grey and green lines in Figure 4.6b are much more similar in length compared to Figure 4.6a, corresponding to the improved tightness of the empirical pairwise bounds.

There is an obvious trade-off between the theoretical pairwise bounds and empirical pairwise bounds: the complexity of calculating the theoretical bounds is constant for each cluster pair, whereas the complexity of finding the empirical bounds is $\mathcal{O}(|C_i \times C_j|)$. However, pairs of clusters occur in multiple cluster combinations. This allows us to cache $l(C_i, C_j)$ and $u(C_i, C_j)$ for later use; in the example of Figure 4.4b, for CC_2, only the empirical bounds for (C_1, C_6) and (C_3, C_6) would have to be calculated, as the bounds of (C_1, C_3) are still cached from CC_1. Caching becomes more effective for higher-cardinality correlation patterns, as more pairwise bounds can be reused.

The empirical bounds take into account the geometrical shape of the clusters by finding the true closest and farthest apart vectors, instead of using the hypothetical worst-case scenario. Therefore, the empirical bounds are typically much tighter; in Figure 4.7, two histograms of pairwise correlations between two clusters' materializations as well as the empirical and theoretical bounds collected during the execution of CD are depicted. The tighter empirical bounds result in less recursions of Algorithm 3. Consequently, the total execution time of CD with empirical bounds is typically an order of magnitude faster compared to CD with theoretical bounds. As such, unless otherwise mentioned, from here on all results will be based on the empirical bounds, rather than the theoretical ones.

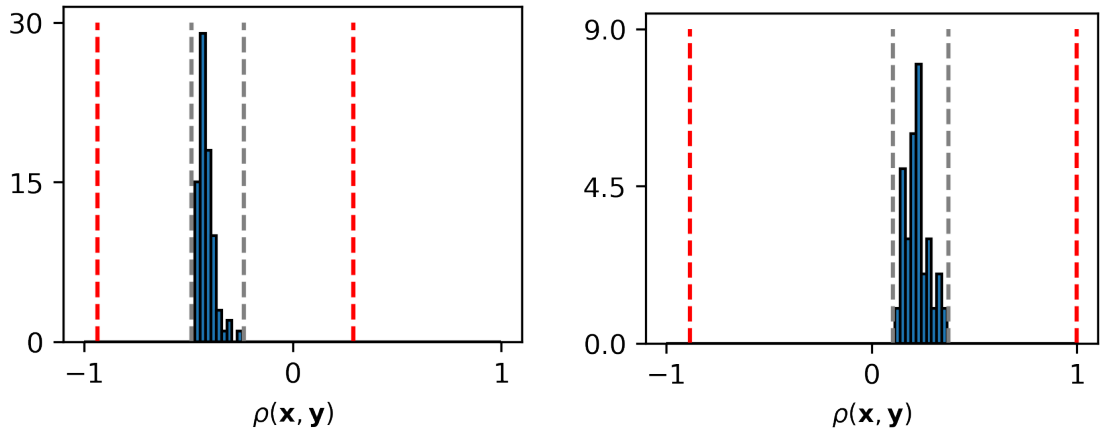


Figure 4.7: Two histograms of correlations between all materializations $(\mathbf{x}, \mathbf{y})$ of a cluster pair (C_i, C_j), as well as theoretical bounds (red) and empirical bounds (grey), found during the execution of CD.

4.5 Extensions for Different Query Types

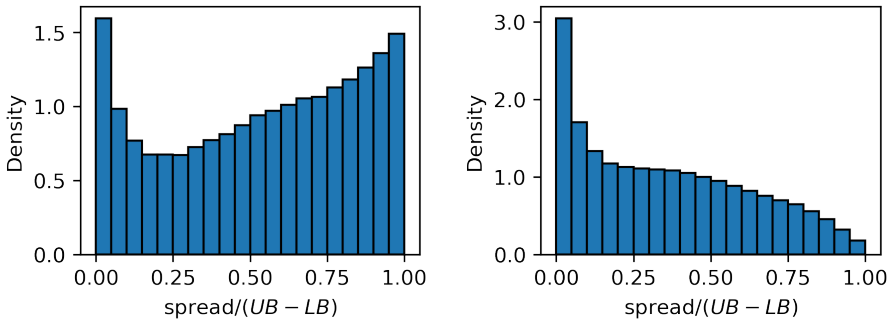
In this section, the extensions to the core CD algorithm for handling progressive and top-k queries are presented. To this end, we first present an extension that changes the exact CD algorithm into an approximation algorithm, which can be tuned to increase performance at the cost of completeness of the result set. The approximation variant of CD serves as a stepping stone for the subsequently presented progressive version, which prioritizes the most promising regions of the search space. Consequently, the progressive variant finds the majority of the results in a comparatively small part of the total execution time, and guarantees a complete result set when its execution is finished. Finally, we show how the progressive variant of CD can be adapted to effectively answer top-k queries.

Approximation Queries

The core CD algorithm as presented in Algorithm 3 guarantees completeness of the result set. However, depending on the application, completeness of the result set may not be a requirement, and the user may simply be interested in finding *some* strongly correlated vector sets, instead of *all* strongly correlated sets. To this end, we modify CD to an approximation variant which can be tuned to trade-off performance for the number of recovered results.

The intuition behind the approximation mechanism is straightforward. When CD uses one of Theorems 4.1 or 4.2 to calculate the bounds for a cluster combination, we observe that these bounds typically have some slack, i.e. the upper bound is generally somewhat higher than the highest correlated materialization of the CC, while the lower bound is generally somewhat lower than the lowest correlated materialization. Figure 4.8 depicts a normalized histogram of the *spread* in correlations of CCs, defined as the highest correlation minus the lowest correlation of any materialization of the CC, divided by $(UB - LB)$. As can be seen in the figure, it is no exception for the spread in correlations to be smaller than $UB - LB$.

The above observation gives rise to a way to adapt CD into an effective approximation algorithm: instead of calculating the real bounds from Theorems 4.1 and 4.2, we assume that there is some slack in these bounds and optimistically tighten the bounds. More specifically, the upper bound is artificially decreased. This might result in CD making a mistake and accidentally discarding a CC of which a materialization is correlated stronger than τ, but based on Figure 4.8, moderate



(a) **mp**(3) (b) **mp**(4)

Figure 4.8: Two normalized histograms of the ratio of the spread of **mp** correlations in a CC divided by $UB - LB$, as collected during the execution of CD.

approximations will only result in limited errors made by accidentally discarding a CC. Note that we do not artificially increase the lower bound, as the correlation of the vast majority of candidates will be smaller than τ. Therefore, artificially increasing the lower bound will only marginally increase performance, while introducing the possibility of false positives.

The shrinking of the upper bound is done as follows: let $\mu = \frac{LB+UB}{2}$ be the *center* of the bounds. Then, for a user-chosen *shrink factor* $\gamma \in [0, 1]$, the shrunk upper bound UB_{shrunk} is calculated:

$$UB_{\text{shrunk}} = \mu + \gamma \cdot (UB - \mu) \tag{4.2}$$

A small γ will result in a drastic approximation, while $\gamma = 1$ results in UB_{shrunk} being equal to the real upper bound. The approximation variant of CD is very similar to the standard version of Algorithm 3. First, the lower and upper bounds are calculated (line 1 of Algorithm 4), as well as UB_{shrunk} (line 3). Then, if $LB > \tau$, the CC is decisive positive, and the results are added to the result set (line 5). If $UB < \tau$, the CC is decisive negative and discarded (line 7). If $UB_{\text{shrunk}} < \tau \leq UB$, the CC is *approximately negative*, and discarded as well (line 9); we explicitly distinguish between decisive negative and approximately negative CCs to facilitate the discussion of the progressive version of CD later in this section. Finally, if $LB < \tau \leq UB_{\text{shrunk}}$, the CC is indecisive, and the algorithm recurses analogous to Algorithm 3 (lines 11-15 of Alg. 4).

Implementation details. In our initial experiments, we found that Algorithm 4 can sometimes result in large parts of the search space being erroneously discarded. To prevent this from happening, our implementation replaces γ by 1 if the geometric mean of the radii of the clusters in the CC is larger than the Euclidean distance corresponding to a correlation of 0.5:

$$\left(\prod_{i=1}^{p} C_i.\text{radius}\right)^{\frac{1}{p}} > \sqrt{2 \cdot d \cdot (1 - 0.5)} \implies \text{replace } \gamma \text{ by 1 for this call of Alg. 4.}$$

This threshold prevents CD from accidentally discarding very large cluster combinations; if the radii of the CC under consideration are too large, there is too much uncertainty about the correlations of the materializations of this CC, and discarding it is too risky. Replacing γ by 1 for this call ensures that $UB_{\text{shrunk}} = UB$, and as such, the true upper bound is used. The following recursive calls of the algorithm again check the geometric mean threshold and apply the value of γ provided by the user if the radii are not too large. The exact value of the threshold was chosen based on exploratory experiments that indicated that this value resulted in a negligible increase in runtime, while significantly improving the algorithm's recall. Furthermore, these experiments showed that

Algorithm 4: ApproximateThresholdQuery($\mathcal{X}$, $\mathcal{Y}$, $Corr$, τ, γ)
Input: Multisets of clusters $\mathcal{X}$ and $\mathcal{Y}$, correlation measure $Corr$, correlation threshold τ, shrink factor γ.

1 $(LB, UB) \leftarrow$ CalcBounds$(\mathcal{X}, \mathcal{Y}, Corr)$ // As in Section 4.4
2 $\mu = \frac{LB+UB}{2}$
3 $UB_{\text{shrunk}} \leftarrow \mu + \gamma \cdot (UB - \mu)$
4 **if** $LB \geq \tau$ **then**
5 | Add $(\mathcal{X}, \mathcal{Y})$ to the result set
6 **else if** $UB < \tau$ **then**
7 | Discard $(\mathcal{X}, \mathcal{Y})$ as decisive negative CC
8 **else if** $UB_{shrunk} < \tau$ **then**
9 | Discard $(\mathcal{X}, \mathcal{Y})$ as approximately negative CC
10 **else**
| // Replace largest cluster with subclusters and recurse
11 | $C_{max} \leftarrow \arg\max_{C \in \mathcal{X} \cup \mathcal{Y}} \{C.radius\}$
12 | Set $SC \leftarrow C_{max}.subclusters$
13 | **for** $S \in SC$ **do**
14 | | $(\mathcal{S}'_l, \mathcal{S}'_r) \leftarrow (\mathcal{X}, \mathcal{Y})$ with C_{max} replaced by S
15 | | ApproximateThresholdQuery$((\mathcal{S}'_l, \mathcal{S}'_r), Corr, \tau, \gamma)$

using the geometric mean resulted in slightly better performance compared to the arithmetic mean.

Progressive Queries

We now discuss how Algorithm 4 can serve as the basis for a progressive algorithm. The main idea behind the approach is as follows: when encountering a CC for which $UB_{\text{shrunk}} < \tau \leq UB$, line 9 of Algorithm 4 will be executed. However, instead of discarding the approximately negative CC, it should postponed for later evaluation with priority based on its 'promisingness', that is, the likelihood that one or more of its materializations' correlations exceeds τ.

To accomplish the above, we present the progressive version of CD in Algorithm 5. First, after running Algorithm 4 (from line 1 of Algorithm 5), the *critical shrink factor* γ^* is calculated for each approximately negative CC as follows (line 13):

$$\gamma^* = \frac{\tau - \mu}{UB - \mu} \tag{4.3}$$

γ^* is the lowest shrink factor γ for which UB_{shrunk} would exceed τ, and as such, for which the CC would not be decisive anymore. Intuitively, a low value of γ^* corresponds to a promising CC: even a very drastic approximation would still lead to Algorithm 4 further examining the CC and splitting a cluster into subclusters. Conversely, a high value of γ^* corresponds to a less promising region of the search space, as even a very mild approximation would lead to the discarding of the CC by Algorithm 4. Note that $\gamma < \gamma^* < 1$.

After running Algorithm 4, ideally, the approximately negative CCs would be processed in order of increasing γ^*. However, constructing a priority queue would take $\mathcal{O}(A \log A)$, where A is the number of approximated CCs, and accessing of the data structure in parallel by multiple threads could result in cores waiting for another thread to unlock the queue. As such, we opt to assign each CC to the i'th out of B priority buckets using the following rule (lines 14-15):

$$i = \lceil \gamma^* \cdot B \rceil \tag{4.4}$$

Algorithm 5: ProgressiveThresholdQuery($\mathcal{X}$, $\mathcal{Y}$, $Corr$, τ, γ, B, scheme)
Input: Multisets of clusters $\mathcal{X}$ and $\mathcal{Y}$, correlation measure $Corr$, correlation threshold τ, initial shrink factor γ, number of buckets B, approximation scheme.

```
1  ApproxNegativeCCs ← ApproximateThresholdQuery(X, Y, Corr, τ, γ)
2  AssignToBuckets(ApproxNegativeCCs, B)
3  for i ∈ 1, ..., B do
4      for CC ∈ bucket i do
5          if scheme = simple then
6              ThresholdQuery(CC, Corr, τ)
7          else
8              γ ← i/B
9              ApproxNegativeCCs ← ApproximateThresholdQuery(X, Y, Corr, τ, γ)
10             AssignToBuckets(ApproxNegativeCCs, B)
11 Method AssignToBuckets(CCs, B):
12     for CC ∈ CCs do
13         γ* ← (τ − μ)/(UB − μ)
14         i ← ⌈γ* · B⌉
15         Put CC in bucket i
```

For sufficiently large B, the CCs will be approximately sorted on γ^*, and each bucket can straightforwardly be processed in parallel as there is no order between CCs within each bucket. We propose two different *schemes* for processing the buckets in order:

- *Simple scheme:* in this scheme, the standard threshold query as described in Algorithm 3 is called, and initialized to start with a CC from the bucket that is currently being processed (line 6). When all CCs from the bucket are completed, the same is repeated for the next bucket, and so on until all buckets are processed.
- *Incremental scheme:* instead of using Algorithm 3, a new value of γ is calculated as follows: $\gamma = \frac{i}{B}$, where i is the index of the bucket being processed (line 8). Subsequently, Algorithm 4 is called again with the new value of γ, and newly found approximately negative CCs are added to the existing buckets based on their value of γ^* (lines 9-10). Note that $\gamma = \frac{i}{B}$ ensures that each CC in the bucket will be indecisive, and that newly found approximately negative CCs will be added to buckets with index higher than i, which will be processed later.

The advantage of the simple scheme is that CCs are assigned to buckets only once, which in practice reduces computational and memory overhead. However, the incremental scheme continuously refines the prioritization of the search space. When a cluster in a parent CC is split, and one of the new child CCs turns out to be less promising, the part of the search space that it represents is postponed and more promising parts of the search space can be examined first. However, this requires adding the child CC to a bucket, incurring additional overhead.

Implementation details. The overhead of getting the parent CC from the bucket and putting its child CCs in buckets can become significant if the child CC just moves one or two buckets from the bucket of its parent. To prevent this issue, our implementation instead uses $\gamma = \min\left(\frac{i}{B} + \beta, 1\right)$ in line 8 of Algorithm 5, where β is a small buffer that ensures the benefit of postponing the CC is significant enough compared to the overhead costs. After exploratory experiments with various datasets and configurations, we found $\beta = 0.1$ to be a good value, and stick to this for the remainder of this work.

Algorithm 6: TOP-KQUERY($\mathcal{X}$, $\mathcal{Y}$, $Corr$, τ, γ, B, scheme)
Input: Multisets of clusters $\mathcal{X}$ and $\mathcal{Y}$, correlation measure $Corr$, desired result set size k, initial shrink factor γ, number of buckets B, approximation scheme.

1 $\mathcal{R}, \tau \leftarrow$ UPDATERESULTSANDTHRESHOLD(pairwiseCorrs, k)
2 ApproxNegativeCCs $\leftarrow$ APPROXIMATETHRESHOLDQUERY($\mathcal{X}, \mathcal{Y}, Corr, \tau, \gamma$)
3 ASSIGNTOBUCKETS(ApproxNegativeCCs, B)
4 **for** $i \in 1, \ldots, B$ **do**
5 **for** $CC \in$ *bucket* i **do**
6 **if** *scheme = simple* **then**
7 THRESHOLDQUERY(CC, $Corr$, τ)
8 **else**
9 $\gamma \leftarrow \frac{i}{B}$
10 ApproxNegativeCCs $\leftarrow$ APPROXIMATETHRESHOLDQUERY($\mathcal{X}, \mathcal{Y}, Corr, \tau, \gamma$)
11 ASSIGNTOBUCKETS(ApproxNegativeCCs, B)
 // update after each bucket as each bucket is processed in parallel
12 $\mathcal{R}, \tau \leftarrow$ UPDATERESULTSANDTHRESHOLD($\mathcal{R}, k$)

13 **Function** `UpdateResultsAndThreshold(`$\mathcal{R}$, k`):`
14 $\mathcal{R} \leftarrow$ SORTDESCENDING($\mathcal{R}$)$[1:k]$
15 $\tau \leftarrow \min_{(X,Y)\in\mathcal{R}} Corr(X, Y)$
16 **return** $\mathcal{R}, \tau$

Top-k Queries

When performing exploratory analyses on an unknown dataset, choosing a suitable correlation threshold τ can be a non-trivial task. A too low value may result in millions of results (and lower efficiency of CD), whereas setting a too high value may lead to spending significant computation time while no results at all are found. Top-k queries are suitable to address this issue, by setting the desired size of the result set instead of a correlation threshold. The answer of the query contains the k highest correlated vector sets that satisfy the correlation pattern and all applicable constraints.

If we had an oracle at our disposal that could predict the precise value of τ that would lead to k results, we could simply run Algorithm 3 to find the k highest correlated results. Of course, such an oracle does not exist. Therefore, many top-k algorithms, such as Fagin's threshold algorithm [16], rely on initializing the algorithm with a conservative threshold, progressively increasing τ based on the answers found so far. As CD depends on τ for effective pruning, increasing the threshold will lead to a more efficient evaluation of the remaining part of the search space. The challenge then becomes (1) how to initialize τ, and (2) how to increase τ as quickly as possible, such that the majority of the execution profits from improved efficiency using the higher value of τ.

The initialization is straightforward. The top-k variant of CD guarantees to return exactly the k highest correlated sets by initializing the threshold τ to the k'th highest pairwise correlation. Note that computation of all pairwise correlations was anyway necessary for determining the pairwise empirical bounds. Then, in order to increase τ as rapidly as possible, the progressive variant is used. After each priority bucket, the result set and correlation threshold τ are updated. Since Algorithm 5 prioritizes the cluster combinations that are likely to be highly correlated, τ can be increased very quickly and often converges to a value close to its final value after only a few buckets. As such, the majority of the execution profits from the high τ for effective pruning of large cluster combinations. For completeness, we present the complete pseudocode of the top-k variant in Algorithm 6, but note that it only differs from Algorithm 5 in lines 1, 12 and 13-16.

4.6 Handling Optional Constraints

Recall the two optional constraints supported by CD from Section 2.2. The irreducibility and minimum jump constraints can straightforwardly be integrated in the CD pruning mechanism. Assume CD is considering a cluster combination $(\mathcal{X}, \mathcal{Y})$ with bounds LB, UB, and that the irreducibility constraint applies. Let $(\mathcal{X}', \mathcal{Y}')$ be a *subset cluster combination*, i.e. $\mathcal{X}' \subseteq \mathcal{X}$, $\mathcal{Y}' \subseteq \mathcal{Y}$ and $(\mathcal{X}', \mathcal{Y}') \neq (\mathcal{X}, \mathcal{Y})$. If $(\mathcal{X}', \mathcal{Y}')$ has a lower bound of at least τ, then we know that for any materialization X, Y of $(\mathcal{X}, \mathcal{Y})$, there exists a (X', Y') s.t. $X' \subseteq X, Y' \subseteq Y, (X', Y') \neq (X, Y)$ and $Corr\,(X', Y') \geq \tau$. Therefore, $(\mathcal{X}, \mathcal{Y})$ can safely be discarded. A similar reasoning holds for the minimum jump constraint: if the lower bound on $(\mathcal{X}', \mathcal{Y}')$ is greater than $UB - \delta$, then no materialization of $(\mathcal{X}, \mathcal{Y})$ can possibly satisfy the minimum jump constraint.

However, the process of evaluating all possible $(\mathcal{X}', \mathcal{Y}')$ would cost $\mathcal{O}(2^{|\mathcal{X}|+|\mathcal{Y}|})$. To avoid this, instead, we opt to only consider the pairwise correlations: note that lower bounds on the pairwise cluster materializations already need to be calculated as input to Theorems 4.1 and 4.2. Specifically, for two-sided metrics, if there exist $C_x \in \mathcal{X}, C_y \in \mathcal{Y}$ such that $l(C_x, C_y) \geq \tau$, then the irreducibility constraint can never be satisfied; if $l(C_x, C_y) \geq UB - \delta$, then the minimum jump constraint can never be satisfied. For one-sided metrics, we apply the same logic to all cluster pairs $C_i, C_j \in \mathcal{X}$.

Considering only the pairwise bounds offers an elegant way to increase the pruning power of CD by exploiting additional constraints, and can result in a notable performance increase depending on the parameters. However, this might still result in false positives in the result set. Consider the example where $Corr(\mathbf{x}, \mathbf{y}) = 0.7$, $Corr(\mathbf{x}, \mathbf{y}, \mathbf{z}) = 0.8$ and $Corr(\mathbf{x}, \mathbf{y}, \mathbf{z}, \mathbf{v}) = 0.9$, with irreducibility constraint and $\tau = 0.75$. Based on the pairwise correlations, $\{\mathbf{x}, \mathbf{y}, \mathbf{z}, \mathbf{v}\}$ would be added to the result set, but there exists a subset of these vectors of which the correlation already exceeds τ, and the irreducibility constraint is violated. To resolve this, CD applies a simple post-processing step in which all materializations of the decisive positive CCs are extracted, and the ones that do not satisfy a constraint are removed. As the size of the result set is expected to be comparably small in practice (in the order of ten thousands of results at most), the post-processing step takes negligible time of the total execution of CD.

Chapter 5

Experimental Evaluation

In this chapter, we examine the performance of CD. The goals of the evaluation are twofold: first, we evaluate how CD's performance changes as a function of the query parameters τ, k, constraints and correlation pattern. Second, we compare CD to an exhaustive search baseline algorithm and the CoMEtExtended algorithm of [3], which is the state-of-the-art for multivariate linear correlation discovery.

We start by describing the setup of the experiments in Section 5.1. We then examine the influence of query parameters on performance in Section 5.2. Finally, we compare CD to a baseline algorithm and CoMEtExtended in Section 5.3.

5.1 Experimental Setup

Hardware and Implementation

All of our experiments are conducted on a single node of the TU/e High Performance Computing cluster. This node is equipped with an Intel Xeon Platinum 8260 processor with 24 cores (48 hyper-threads) clocked at 2.4 GHz with 380 GB of RAM for each experiment.

CD is implemented in Java 8 [49] and uses the Java Stream API to parallelize the computation. For the multipole metric, computation of eigenvalues and matrix norms is necessary, for which we use the open-source Jama library [29]. The multi-threaded implementation of CoMEtExtended was kindly provided to us by the authors of [3]; we only made minor and straightforward modifications to this implementation such that it can use all cores available to the machine.

Datasets

A variety of datasets is used to examine the performance of CD. At the time of writing, all data is publicly available.

- *Stocks:* the stocks data consists of prices of securities measured at 5 minute intervals, measured from April 1, 2020 to May 12, 2020. Missing values are imputed using simple linear interpolation. This results in a data set of 1596 stocks with 9103 observations for each time series. Every time series is preprocessed by computing the log-returns of the prices, as is commonly done in finance. The names of the stocks in our dataset can be found in [39], and the data itself is freely available from sources like Yahoo!Finance.
- *fMRI:* this dataset contains functional Magnetic Resonance Imaging data of a participant watching the movie "500 Days of Summer" [4]. We used the file found under `derivatives/sub1/func/sub-1_task-500daysofsummer_bold_blur_censor`, which is already preprocessed for voxel-based analytics. The data has a resolution of 64x76x64 voxels and 5470

timestamps. We further preprocess the data by mean-pooling the voxels with kernels of size 2x2x2, 3x3x3, 4x4x4, 6x6x6 and 8x8x8 voxels, resulting in 5 datasets. Time series with constant activity level (i.e. $\sigma = 0$) are removed after mean-pooling. The resulting datasets contain 9700, 3152, 1440, 509 and 237 time series respectively, each with 5470 observations. Unless otherwise mentioned, the dataset with 1440 time series (resulting from 4x4x4 mean pooling) is used when referring to the fMRI data.

- *SLP:* the SLP dataset [31] was one of the datasets used for evaluating CoMEtExtended in [3], and as such is a logical choice for comparing CD to CoMEtExtended with regard to the task of discovering strong multipole correlations. We use the already preprocessed version that is used in [3] (again, we thank the authors for their kindness in sharing the data). The SLP data contains 171 aggregated Sea Level Pressure time series with 108 observations each. We refer to [3] for the details on dataset characteristics and preprocessing.
- *ISD:* the ISD dataset [48] contains weather data collected from sensors across the globe between 2000 and 2005. We limit ourselves to the temperature (ISD-temp) and Sea Level Pressure (ISD-SLP) data. For each of the two datasets, time series which had missing values for more than five consecutive days are removed, and sensor output was resampled to 6 hour intervals. Missing values are imputed using simple forward filling. This results in ISD-temp containing 2927 time series and ISD-SLP containing 1898 time series, all with 8760 observations.
- *Crypto:* the Crypto data contains 3 hour granularity prices of various cryptocurrencies, collected via the CoinGecko API [12]. The preprocessing is identical to the stocks dataset. The result is a dataset containing 3937 cryptocurrencies with 712 observations for each, collected between April 14, 2020 and July 13, 2020. Details on getting the data can be found at [39].

We consider stocks and fMRI as *main* datasets, for which a thorough evaluation is performed. The other datasets serve the purpose of confirming our findings via a more superficial analysis, or in the case of SLP, for comparing to CoMEtExtended. A summary of the datasets and their most important characteristics can be found in Table 5.1, where $|\bar{\rho}|$ is the mean of the absolute pairwise Pearson correlations in the data.

	Stocks	fMRI (default resolution)	SLP	ISD-temp	ISD-SLP	Crypto
n	1596	1440	171	2927	1898	3937
d	9103	5470	108	8760	8760	712
$\lvert\bar{\rho}\rvert$	0.036	0.060	0.076	0.289	0.135	0.043

Table 5.1: Main characteristics of datasets

5.2 Influence of Query Parameters

First, it his examined how query parameters like τ, k, the progressive scheme and the correlation pattern affect performance. For conciseness, the clustering parameters K, ϵ_{start}, α and $h_{\max}$ and prioritization parameters γ and B are not varied here; we found that, as long as these are being set within a reasonable range, their effect on performance was small. A table with experiments that investigate the effect of the clustering parameters can be found in Appendix A, while the parameters used for the experiments can be found in table 5.2.

Influence of τ, k and constraints

First, the effect of crucial query parameters like τ, k and additional constraints on performance is examined. For threshold queries, our implementation was configured to automatically switch

Parameter	ϵ_{start}	K	α	$h_{\max}$	n_repetitions	γ	B
Value	$\sqrt{0.5 \cdot d \cdot (1 - 0.25)}$	10	0.8	20	50	0	50

Table 5.2: Values of fixed parameters.

to a top-k query as soon as 10 000 000 results are found – finding even more results is not useful in practice and comes with significant overhead for simply storing the solution set, resulting in distorted computation times. Therefore, the computation times for these queries are discarded and not shown in this section. Each query was run 25 times. We found that execution times were influenced by external noise (e.g. due to the OS), and therefore, to get a clearer picture of the relationship between query parameters and performance, the three lowest and three highest execution times of each query are discarded. The average was calculated for the remaining execution times.

Effect of τ. Figures 5.1 and 5.2 show the influence of τ and constraints on the performance of CD, for different correlation patterns and the fMRI and stocks data respectively. Overall, the figures show the same intuition: increasing τ leads to better performance and, depending on the other parameters, adding an additional constraint can boost performance as well. The effect of τ was to be expected: a higher value of τ will result in $UB < \tau$ for larger cluster combinations, thereby enabling CD to prune the search space more effectively. Also, note that the multipole correlation requires significantly more computational effort; this is because computing its bounds requires calculating eigenvalues and matrix norms, which takes $\mathcal{O}(l_{\max}^3)$. Moreover, because **mp** by definition finds optimally weighted vector combinations, its correlation values tend to be higher, and therefore CD's pruning power is less strong than for **mc**.

The magnitude of the effect of optional constraints differs per query. Consider, for example, Figures 5.2c and 5.1c. For $\tau = 0.825$, adding a constraint to $\mathbf{mc}(2, 2)$ on the stock data (Fig. 5.2c) reduces the execution time by as much as 51%, whereas for $\mathbf{mc}(2, 2)$ on fMRI (Fig. 5.1c) execution time decreases with only 13.4%. We found that the magnitude of the constraints' effect on performance correlates with the reduction in the size of the result set that is found. Illustratively, for $\tau = 0.825$, removing a jump constraint of $\delta = 0.1$ from $\mathbf{mc}(2, 2)$ on the stocks data multiplies the size of the result set by 10087 (Fig. 5.2c); doing the same for $\mathbf{mc}(2, 2)$ on fMRI multiplies the amount of results by only 147 times (Fig. 5.1c).

Effect of k. Figures 5.3 and 5.4 depict the results for top-k queries on fMRI and stock data. As the irreducibility constraint does not apply to top-k queries (recall Section 2.2), we evaluate $\delta = 0.05$ and $\delta = 0.1$ as minimum jump constraints. For brevity, we only consider the simple progressive scheme. As expected, increasing k typically leads to an increase in execution time; this is because the rolling value of τ can be increased quicker when k is small. As is also the case for threshold queries, a high τ leads to more effective pruning.

In contrast to threshold queries, adding a constraint generally *increases* computation time for top-k queries. At first sight, this appears counterintuitive: as described in Section 4.6, adding a constraint may enable CD to prune a cluster combination that otherwise would need to be recursed upon. Again, this observation can be attributed to the difference in the rolling value of τ. A constraint may result in a very highly correlated vector set being discarded, for example, because there is a very high pairwise correlation in the set already. As such, the correlation of this vector set cannot contribute to the k highest correlations, and the rolling value of τ is lower.

The same intuition also explains why execution time for queries without constraints generally grows less fast with increasing k compared to their counterparts with minimum jump constraints. Without constraints, there are much more highly correlated vector sets that can be considered, and as such, the 500'th highest correlation value does not differ much from the 100'th highest cor-

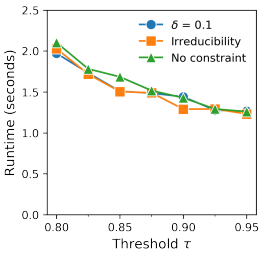
relation value. When a constraint is applied, the set of feasible candidates is smaller, and varying k leads to a significant difference in the rolling value of τ.

To illustrate the influence of constraints and k on the rolling value of τ and computation time, consider **mp**(4) on fMRI (Fig. 5.3e). Table 5.3 shows the k'th highest correlation value τ_k of the result set and execution times for different δ and k. Clearly, the amount with which τ_k decreases as k increases strongly correlates with the execution time.

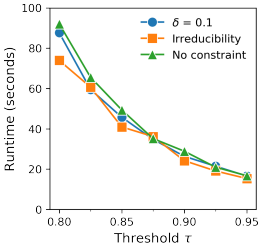
Interestingly, **mc**$(2, 2)$ on fMRI (Fig. 5.3c) is the only query in which runtimes for $\delta = 0.05$ are generally lower than for queries without a constraint. On a closer look, this can also be explained by looking at the value of τ: the difference in τ_k between $\delta = 0.05$ and no constraint queries differs by less than 0.012 for all values of k. In contrast, for **mc**$(2, 2)$ on stocks (Fig. 5.4c), the difference ranges from 0.03 to 0.06, and for **mc**$(1, 3)$ on fMRI (Fig. 5.3b) from 0.045 to 0.05. Apparently, the effect of the marginally higher rolling τ is smaller than the pruning power of the minimum jump constraint for **mc**$(2, 2)$ on fMRI (Fig. 5.3c). On the other hand, for **mc**$(2, 2)$ on fMRI with $\delta = 0.1$, the difference in τ_k is at least 0.05 with the no-constraint query, explaining why the difference in execution time between these two queries is in line with the other top-k queries of Figures 5.3 and 5.4.

Constraint	None		$\delta = 0.05$		$\delta = 0.1$	
k	Time (s)	τ_k	Time (s)	τ_k	Time (s)	τ_k
100	513.5	0.917	645.6	0.865	812.4	0.825
200	516.3	0.916	696.0	0.852	944.6	0.802
300	518.6	0.916	739.1	0.845	1026.9	0.792
400	515.1	0.915	753.9	0.839	1089.1	0.783
500	518.3	0.915	785.0	0.835	1155.2	0.777

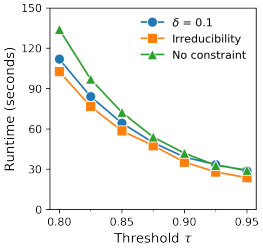
Table 5.3: Execution times and τ_k for **mp**(4) on fMRI.



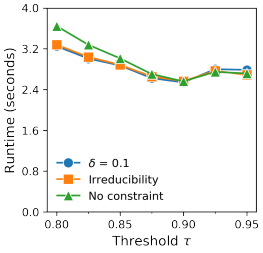
(a) **mc**$(1, 2)$



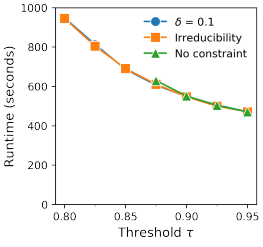
(b) **mc**$(1, 3)$



(c) **mc**$(2, 2)$

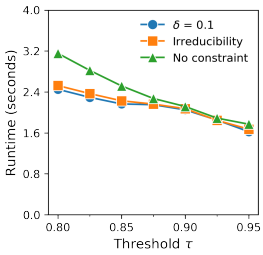


(d) **mp**(3)

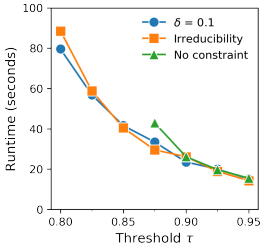


(e) **mp**(4)

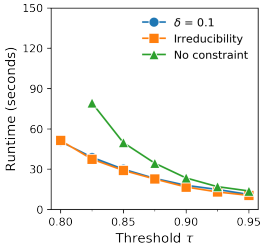
Figure 5.1: fMRI threshold queries



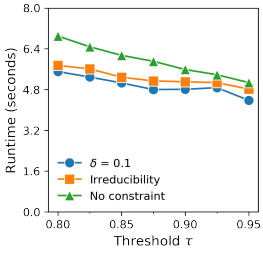
(a) **mc**$(1, 2)$



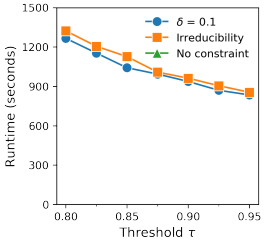
(b) **mc**$(1, 3)$



(c) **mc**$(2, 2)$

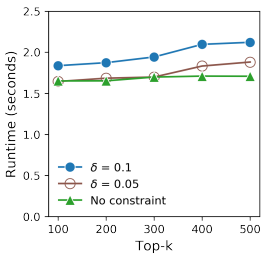


(d) **mp**(3)

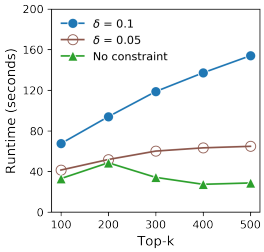


(e) **mp**(4)

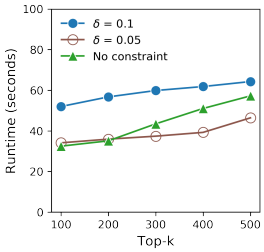
Figure 5.2: Stocks threshold queries



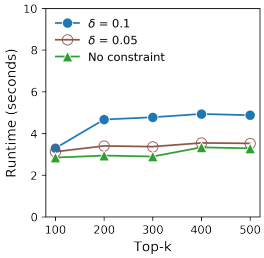
(a) **mc**$(1, 2)$



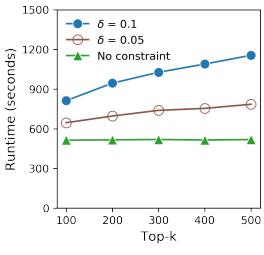
(b) **mc**$(1, 3)$



(c) **mc**$(2, 2)$

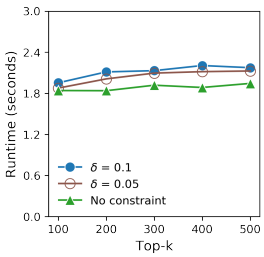


(d) **mp**(3)

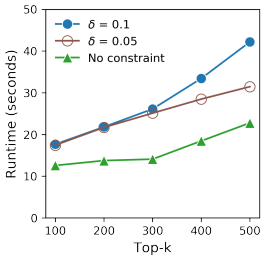


(e) **mp**(4)

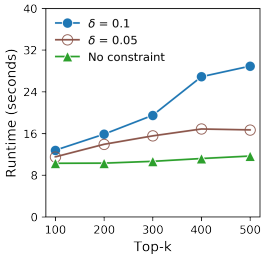
Figure 5.3: fMRI top-k queries



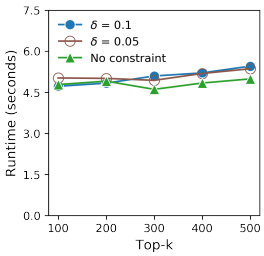
(a) **mc**$(1, 2)$



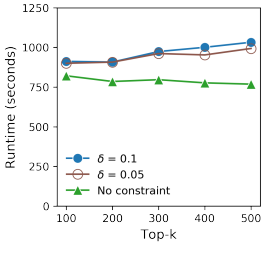
(b) **mc**$(1, 3)$



(c) **mc**$(2, 2)$



(d) **mp**(3)



(e) **mp**(4)

Figure 5.4: Stocks top-k queries

Influence of Correlation Pattern

Table 5.4 shows the results for varying correlation patters for $\tau = 0.9$ and a jump constraint of $\delta = 0.05$ on both main datasets; each query was set to time out after 20 hours to prevent occupying shared resources for an excessive amount of time. Note that the increase in execution time of CD is typically around two orders of magnitude less compared to the increase in search space size. For example, comparing $\mathbf{mc}(1, 2)$ and $\mathbf{mc}(1, 4)$ on the stocks dataset, the search space grows five orders of magnitude, whereas the execution time of CD increases with approximately three orders of magnitude – a considerable amount, but significantly less than what is expected for a baseline approach that iterates over all candidates. Similarly, comparing $\mathbf{mc}(1, 2)$ and $\mathbf{mc}(2, 3)$ for fMRI, the search space increases with five orders of magnitude, whereas computation time increases with four orders of magnitude. Of course, due to the explosion of the search space (#candidates column in Tab. 5.4) inherent to correlation patterns of increasing complexity, CD also has its limits. Query patterns with $l_{\max} + r_{\max} = 5$ at most could be completely evaluated within the specified time.

	Stocks			fMRI		
	Runtime (s)	#results	#candidates	Runtime (s)	#results	#candidates
$\mathbf{mc}(1, 2)$	2.0	581	$2.03 \cdot 10^9$	1.4	33	$1.49 \cdot 10^9$
$\mathbf{mc}(1, 3)$	23.3	632	$1.08 \cdot 10^{12}$	26.8	45	$7.14 \cdot 10^{11}$
$\mathbf{mc}(2, 2)$	18.2	1875	$8.10 \cdot 10^{11}$	41.5	1045	$5.37 \cdot 10^{11}$
$\mathbf{mc}(1, 4)$	6369.9	646	$4.29 \cdot 10^{14}$	6294.0	45	$2.56 \cdot 10^{14}$
$\mathbf{mc}(2, 3)$	4238.6	2796	$8.58 \cdot 10^{14}$	15760.5	2184	$5.12 \cdot 10^{14}$
$\mathbf{mc}(1, 5)$	> 20 hrs	-	$1.36 \cdot 10^{17}$	> 20 hrs	-	$7.35 \cdot 10^{16}$
$\mathbf{mc}(2, 4)$	> 20 hrs	-	$3.41 \cdot 10^{17}$	> 20 hrs	-	$1.84 \cdot 10^{17}$
$\mathbf{mc}(3, 3)$	> 20 hrs	-	$2.29 \cdot 10^{17}$	> 20 hrs	-	$1.23 \cdot 10^{17}$
$\mathbf{mp}(3)$	4.6	302	$6.76 \cdot 10^8$	2.6	10	$4.97 \cdot 10^8$
$\mathbf{mp}(4)$	966.4	576	$2.69 \cdot 10^{11}$	560.0	12	$1.78 \cdot 10^{11}$
$\mathbf{mp}(5)$	> 20 hrs	-	$8.58 \cdot 10^{13}$	> 20 hrs	-	$5.12 \cdot 10^{13}$

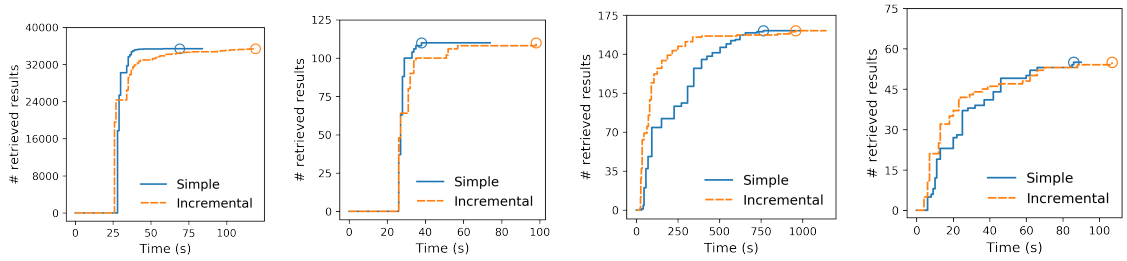
Table 5.4: Results for different threshold queries ($\tau = 0.9$, $\delta = 0.05$).

Progressive Queries

In order to evaluate the effectiveness of the prioritization mechanism of Algorithm 5, we will examine how the amount of discovered results grows during the execution of the query. In all experiments, we will set the initial value of γ to 0; exploratory experiments showed that setting γ to a greater starting value typically has a negligible effect on the total execution time, but delays the speed with which the first results come in significantly. Therefore, setting $\gamma = 0$ as a starting value is best aligned with the objective of the progressive query to yield as many results as quickly as possible.

We stress that for the evaluation of progressive queries, we do not consider all correlation patterns *up to* $l_{\max}, r_{\max}$, but only the pattern $l_{\max}, r_{\max}$ itself. Also considering all smaller patterns would defy the purpose of evaluating the progressive approach, as evaluating smaller-cardinality patterns takes much less time. Since these are considered first, also showing the part of the execution corresponding to the smaller-cardinality patterns would be misleading, as it would show a large amount of results discovered in only a few seconds, which is not the result of the prioritization mechanism.

Figure 5.5 compares the simple and incremental progressive schemes for various queries. The circle on each line indicates the first point in time at which the full result set was recovered; the end of the line indicates the finishing of the execution and is the moment the user is *guaranteed* completeness of the result set. Interestingly, both the simple and the progressive schemes are able



(a) fMRI, $\mathbf{mc}(2,2)$, $\tau = 0.875$.

(b) fMRI, $\mathbf{mc}(2,2)$, $\tau = 0.875$, $\delta = 0.1$.

(c) Stocks, $\mathbf{mp}(4)$, $\tau = 0.875$, $\delta = 0.1$.

(d) Stocks, $\mathbf{mc}(1,3)$, $\tau = 0.8$, $\delta = 0.1$.

Figure 5.5: Development of number of retrieved results over time for various progressive queries. Circles indicate the first time when all results have been found, whereas the end of each line indicates the finishing of Algorithm 5.

to discover a large fraction of the result set in a comparatively small part of the total execution time. Illustratively, in Figures 5.5a, 5.5b and 5.5c, using the simple scheme yields more than 90% of the results in 50% of the total runtime (in Figure 5.5d, 89.1% of the result set is discovered in 50% of the execution time). Furthermore, the figures illustrate the trade-off between the two schemes: the incremental scheme generally yields the first results quicker, but has a 'longer tail' and requires more time to recover all results and finish the execution.

The above observation is the result of how the schemes differ with regard to increasing γ. The incremental scheme gradually increases γ after the completion of each priority bucket, thereby postponing newly found cluster combinations that are less promising to later buckets. Consequently, the most promising cluster combinations can be examined faster in the first buckets, but additional overhead for re-allocating and retrieving cluster combinations to and from later buckets is incurred. In contrast, the simple scheme straightforwardly examines all cluster combinations that were assigned to a bucket during the initialization phase. This results in less overhead, but splitting the cluster combinations might result in new CCs that are not likely to have a materialization satisfying the query. Therefore, the first buckets generally take longer to finish compared to the incremental scheme.

Miscellaneous Datasets

To examine how the dataset properties affect the performance of CD, we repeat the top-100 queries on the miscellaneous datasets. The results (Table 5.5) are in line with earlier findings. Increasing the correlation pattern's complexity from $\mathbf{mc}(1,2)$ to $\mathbf{mc}(1,3)$ or $\mathbf{mc}(2,2)$ increases runtime by one or two orders of magnitude, but the growth in runtime is much less than the increase in search space size (around three orders of magnitude).

Both ISD datasets contain extremely high correlations, reflected in a high value of τ_k. This can be attributed to the spatial locality in the datasets; sensors located close to each other are likely to report similar measurements, and therefore, the (pairwise) correlations in the data are very high. For the crypto data, increasing the correlation pattern does significantly increase the 100'th highest correlation τ_k, indicating that stronger relationships are found in this data by increasing the complexity of the correlation pattern.

	ISD-temp		ISD-SLP		Crypto	
	Time	τ_k	Time	τ_k	Time	τ_k
$\mathbf{mc}(\mathbf{1},\mathbf{2})$	13.0	0.991	3.3	0.998	6.2	0.906
$\mathbf{mc}(\mathbf{1},\mathbf{3})$	60.9	0.993	10.3	0.998	501.0	0.936
$\mathbf{mc}(\mathbf{2},\mathbf{2})$	174.2	0.995	34.9	0.999	303.1	0.942
$\mathbf{mp}(\mathbf{3})$	22.2	0.992	8.4	0.999	30.2	0.938
$\mathbf{mp}(\mathbf{4})$	7223.6	0.993	2952.2	0.999	12462.7	0.952

Table 5.5: Top-100 queries, no-constraints, on ISD and Crypto datasets

5.3 Comparison with Baseline and Existing Work

In order to assess the efficiency of CD, we first compare its execution times to those of a baseline algorithm. Then, we compare CD to CoMEtExtended (and also CoMEt, which is a special case of CoMEtExtended with $\rho_{\text{CE}} = 0$) [3].

Comparison to Baseline and Scalability in n

Baseline algorithm. The multi-threaded baseline algorithm works as follows: first, all pairwise correlations between vectors are calculated and cached. Then, the baseline algorithm enumerates over all vector sets, and computes the correlation value using the cached pairwise correlations. Recall from Section 4.4 that for **mc**, this can be done as follows:

$$\mathbf{mc}(X,Y) = \frac{\sum_{\mathbf{x}\in X}\sum_{\mathbf{y}\in Y}\rho(\mathbf{x},\mathbf{y})}{\sqrt{\sum_{\mathbf{i},\mathbf{j}\in X}\rho(\mathbf{i},\mathbf{j})}\sqrt{\sum_{\mathbf{i},\mathbf{j}\in Y}\rho(\mathbf{i},\mathbf{j})}}$$

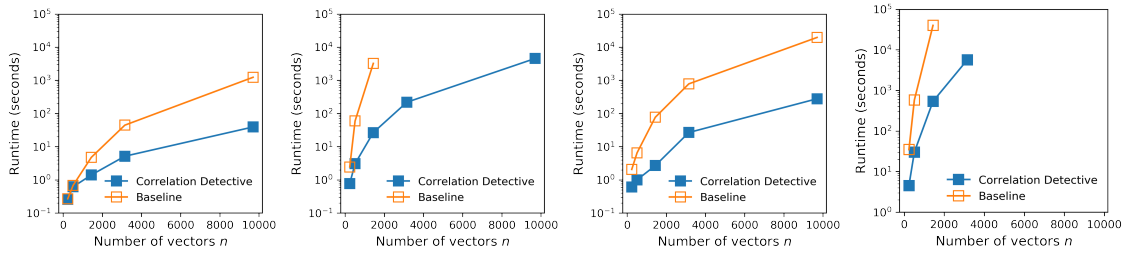
and for **mp** (with pairwise correlation matrix $\mathbf{R}$, as in [3]):

$$\mathbf{mp}(X) = 1 - \lambda_{\min}(\mathbf{R})$$

Note that since the baseline algorithm reuses the precomputed pairwise correlations, its asymptotic complexity is already constant in the dimensionality d of each vector. In addition, we also apply the trivial optimization of not considering symmetrical candidates: if $\mathbf{mc}(\{\mathbf{x}\},\{\mathbf{y},\mathbf{z}\})$ is already considered, we do not consider $\mathbf{mc}(\{\mathbf{x}\},\{\mathbf{z},\mathbf{y}\})$ as it leads to the same correlation, and similar for other correlation patterns.

Scalability of baseline algorithm and CD. To test how well the baseline algorithm and CD scale with the number of vectors n, we run both on the fMRI data with varying resolutions; this corresponds to a realistic domain use-case where a neuroscientist wishes to analyse fMRI images at varying spatial resolutions to obtain insights at different granularity levels of the brain's anatomy. Figure 5.6 shows how the runtime of both CD and the baseline algorithm grows with increasing n, by changing the resolution of the fMRI data. Note that again, queries were set to time out after twenty hours; the timed out queries are left out of the figure.

For all queries, the execution time of CD grows at a considerably slower rate compared to the baseline algorithm. Moreover, the difference between the two algorithms increases as n becomes larger. This suggests that CD is able to prune a larger fraction of the search space for datasets containing more vectors. Intuitively, this can be explained as follows: if a vector is added to the data and this vector belongs belongs to an already existing cluster C, but it does not change the pairwise bounds between C and other clusters in a decisive cluster combination, then CD can prune more candidates with the same computational effort as for the dataset before the added vector. Of course, in general adding a vector to a cluster C may change the bounds of the decisive



(a) **mc**$(1, 2)$ (b) **mc**$(1, 3)$ (c) **mp**(3) (d) **mp**(4)

Figure 5.6: Queries on the fMRI data with varying resolutions ($\tau = 0.9$, no constraints).

cluster combinations that contain C such that they become indecisive. However, even only some decisive CCs remaining decisive can still help in pruning a larger fraction of the search space.

Comparison to CoMEtExtended

Our next experiment focuses on comparing CD with CoMEtExtended. Notice that CoMEtExtended's goal differs slightly from our problem statement. As described in [3], CoMEtExtended aims to find only *maximal* sets that exhibit a strong multipole correlation, whereas CD finds *all* sets (up to a specified cardinality) that are strongly correlated. Furthermore, CoMEtExtended is an approximate algorithm, and as such, may not yield a complete result set. Its recall (and performance) can be controlled by parameter ρ_{CE}, which takes values between -1 and 1; values around 0 offer a reasonable trade-off between performance and recall [3]. In contrast, CD is exact, and therefore always retrieves all answers.

We compare the two algorithms by examining the amount of discovered results and runtimes. To ensure not putting CoMEtExtended at a disadvantage, we also compute the correlation for all *subsets* of the sets found by CoMEtExtended, and include them in the answer set if and only if they satisfy the query. By doing so, CoMEtExtended's recall is increased to a value as if its goal were to also discover non-maximal sets. Note that this is performed as a post-processing step, and the required computation time is not included in the runtime of CoMEtExtended, as to not penalize its performance. Table 5.6 presents the number of results and execution time of CoMEtExtended and CD on the same dataset (SLP) and for the same parameters used in [3]. Notice that we only consider the **mp** measure, since CoMEtExtended does not support **mc**.

	CoMEtExtended						Correlation Detective			
τ, δ	$\rho_{\text{CE}} = 0$		$\rho_{\text{CE}} = 0.01$		$\rho_{\text{CE}} = 0.02$		**mp**(4)		**mp**(5)	
	time	#res.	time	#res.	time	#res.	time	#res.	time	#res.
0.4, 0.1	604	62663	1318	67110	3530	70921	11	71083	899	88305
0.4, 0.15	511	7244	1218	7300	3393	7343	9	7559	575	7562
0.4, 0.2	501	2166	1210	2171	3327	2174	7	2183	333	2183
0.5, 0.1	459	30632	1099	33718	2836	36457	7	34592	557	51391
0.5, 0.15	398	3646	1006	3702	2760	3745	6	3961	391	3964
0.5, 0.2	390	1434	1006	1439	2701	1442	6	1451	292	1451
0.6, 0.1	246	7823	598	8892	1592	9859	5	9204	310	17349
0.6, 0.15	223	1569	577	1606	1559	1635	5	1840	245	1843
0.6, 0.2	219	771	568	776	1532	779	5	788	199	788

Table 5.6: Comparison of CoMEtExtended and Correlation Detective: running time (seconds) and number of retrieved results

CD is consistently faster than CoMEtExtended and often discovers significantly more results. Indicatively, for **mp**(4), CD is one to two orders of magnitude faster than CoMEtExtended with $\rho_{\text{CE}} = 0$, whereas for $\rho_{\text{CE}} = 0.02$, the difference in performance is between two to three orders of magnitude, while CD often finds more results, or at least a comparable amount.

For queries with $\delta = 0.1$, CoMEtExtended found 281 sets of cardinality 6, and one set of cardinality 7 ($\rho_{\text{CE}} = 0.02, \tau = 0.4$). Naturally, these were not discovered by CD, as the queries specified $l_{\max} = 5$ at most, prioritizing the simpler and more interpretable results. Therefore, CoMEtExtended might still be useful for potential applications where completeness of the result set is not required *and* the user is specifically interested in discovering maximum-cardinality vector sets. As noted in [3], the size of the vector sets satisfying the query typically decreases as δ is increased. For all other use cases, CD is the better choice in terms of both efficiency and completeness.

Furthermore, note that the values of τ in Table 5.6 are quite low – the query parameters are taken from [3] in order to replicate their experiments. Interestingly, as the low τ is less restrictive on the result set, the magnitude of δ plays a significant role in the efficiency of CD in particular, since a considerably large part of its pruning power must now come from the constraint-based pruning rules explained in Section 4.6.

Chapter 6

Conclusion and Future Work

6.1 Conclusion

Prior studies have highlighted the great potential of multivariate correlation analysis for aiding scientists, engineers and businesses alike in making data-driven discoveries and decisions [2, 3, 11, 18, 34, 45, 46, 47, 57, 60]. However, the inherent combinatorial explosion of the search space prohibits the application of multivariate correlation analyses *at scale* in practice. Prior works that address this problem are limited in terms of their general applicability as they make assumptions about the data [60], rely on hand-crafted constraints provided by the user [2, 3, 60] or do not guarantee any degree of completeness of the result set [2, 3, 45].

In this thesis, we propose a more generally applicable framework that supports two complementary correlation measures. Correlation Detective guarantees completeness of the result set and does not rely on any additional assumptions on the data and additional constraints. Moreover, we extend the standard threshold query to approximation, progressive and top-k query types, and allow the integration of *optional* constraints that enhance the quality and interpretability of the results. As such, a great degree of flexibility is provided to the user to specify a query that is most suitable for the use-case at hand.

In summary, the evaluation of Correlation Detective shows the following:

- CD exploits the restrictive nature of τ, k and/or constraints on the result set to effectively prune the search space. The more restrictive the query parameters, the higher the pruning power, and consequently efficiency, of CD.
- CD's computation time still increases when the correlation pattern becomes more complex. However, the execution time required for increasingly complex correlation patterns grows at a substantially slower rate than the size of the search space.
- The progressive variant of CD effectively prioritizes the most interesting candidates in order to find the majority of the result set in a comparatively small part of the computation time. Consequently, this variant can also be used as an approximation algorithm that finds as many results as possible within a limited computation time, or that terminates once a sufficient amount of results has been discovered.
- CD scales significantly better with the amount of vectors than a baseline algorithm that iterates over all candidates, resulting in CD being orders of magnitude faster for large datasets.
- CD is orders of magnitude faster than an existing method addressing the **mp** measure, while retrieving more strongly correlated vector sets.

6.2 Future Work

Due to time constraints, not all ideas that arose during the process of working on this thesis could be addressed. In this section, we propose several research directions for extending and improving our methods in future works.

Correlation Measures

It would be interesting to investigate if other correlation measures can be supported by Correlation Detective. The crucial requirement for a correlation measure to be eligible for our algorithm is that bounds on the multivariate correlation value can be derived from bounds on pairwise correlations. The strategy employed in this research to derive such bounds is to rewrite the multivariate correlation as a function of pairwise correlations and bound the function value based on bounds on its inputs. Two interesting measures that are worth looking into are Total Correlation and Canonical Correlation Analysis (CCA).

Total Correlation. Total Correlation might not lend itself to be written as a function of solely pairwise Mutual Informations. Nevertheless, Nguyen et al. [45] effectively approximate the Total Correlation value by constructing a maximum spanning tree of the pairwise Mutual Informations of the vectors, and summing the edges' values. Taking for granted that this will not yield an exact algorithm, it is interesting to investigate if this approach can be integrated with our methods to design an algorithm even more efficient than in [45].

Canonical Correlation Analysis. We already saw in Section 3.1 that the CCA measure can be written as a function of pairwise correlations, and as such, it seems plausible that the value can be bounded in a similar way as **mc** and **mp**. The author did attempt to derive such bounds, but was not able to succeed in the given timeframe. Another possible direction would be to bound the value of $CCA(X, Y)$ by $\mathbf{mp}(X \cup Y)$, as both measures are quite similar in terms of their intuitive interpretation of measuring linear dependence. The **mp** measure is more flexible, as it is not constrained to look for correlations between two non-overlapping vector sets, and as such we expect its correlation to be higher than for CCA. Initial experiments showed that indeed, the value of $\mathbf{mp}(X \cup Y)$ seems to generally exceed the value of $CCA(X, Y)$, or be only slightly less. This is illustrated in Figure 6.1, where we randomly sampled 5000 sets X, Y ($|X| = |Y| = 2$) from the SLP data, and computed the values of $CCA(X, Y)$ and $\mathbf{mp}(X \cup Y)$. However, the author was not able to derive strict guarantees in the timeframe of this thesis. At the time of writing, the author thinks this direction is the most promising to pursue for CCA in future work.

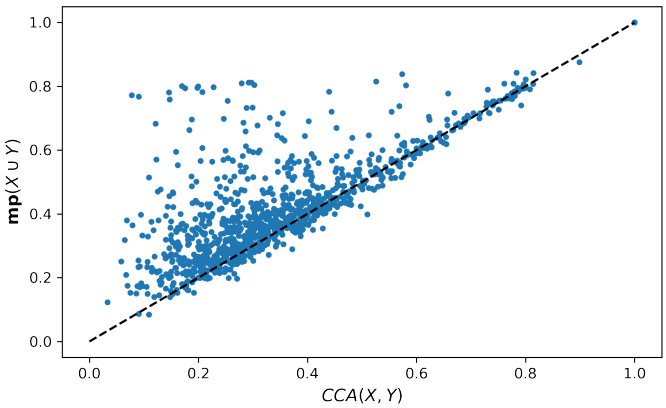


Figure 6.1: $CCA(X, Y)$ and $\mathbf{mp}(X \cup Y)$ for randomly sampled X, Y from the SLP dataset.

Non-linear and Lagged Extensions

Another useful extension to the current work would be to generalize the currently supported correlation measures to also be able to discover non-linear relationships in the data. As the multipole measure is closely related to Principal Component Analysis, for which kernel methods are available [27], it seems plausible that kernel methods are applicable to this measure. Further research must be done to determine whether this is the case.

If kernel methods cannot be applied, it might make sense to add 'artificial' vectors to the dataset. E.g., for every vector $[x_1, x_2, \ldots, x_d]^T$, another vector $[x_1^2, x_2^2, \ldots, x_d^2]^T$ is added to the data in order to discover quadratic relationships. To prevent further combinatorial explosion of the search space, the artificial vectors should not be considered separately, but only if the corresponding real vector is already in (X, Y). For the **mp** measure, this would correspond to extending the correlation matrix $\mathbf{R}$ of which the eigenvalue is computed with the pairwise correlations between artificial vectors. For lagged correlations, a similar strategy might work, in which the lagged vectors are only considered in combination with their non-lagged counterparts.

Collaboration with Domain Experts

In this work, we built upon earlier research that already demonstrated the usefulness of the supported correlation measures in various domains. However, given that CD can handle larger datasets than existing methods, it would be interesting to collaborate with domain experts and discover new applications. As already mentioned in Section 1.2, one parallel master thesis project investigated such an application in the domain of finance, in collaboration with De Nederlandsche Bank. We also discussed with several experts in the field of fMRI image analysis, who were enthusiastic about potential applications of our work, for example in the context of real-time neurofeedback [28, 38, 62].

Distributed Correlation Detective

Another useful project that could greatly enhance the practical value of the proposed methods is that of distributing the discovery of strong multivariate correlations over a cluster, for example using the Apache Spark software package [17]. A well-designed distributed multivariate correlations discovery API would allow organizations, researchers and engineers to easily scale-out their queries in the cloud, adding significantly to the practical usability of the software and enabling the processing of larger data sets. Our current implementation already makes use of parallelization using the MapReduce-like syntax of the Java stream API, and may serve as a good starting point for the distribution of CD over multiple machines.

References

[1] Rakesh Agrawal and Ramakrishnan Srikant. "Fast Algorithms for Mining Association Rules in Large Databases". In: *Proceedings of the 20th International Conference on Very Large Data Bases*. VLDB '94. San Francisco, CA, USA: Morgan Kaufmann Publishers Inc., 1994, pp. 487–499. DOI: 10.5555/645920.672836.

[2] Saurabh Agrawal, Gowtham Atluri, Anuj Karpatne, William Haltom, Stefan Liess, Snigdhansu Chatterjee and Vipin Kumar. "Tripoles: A New Class of Relationships in Time Series Data". In: *Proceedings of the 23rd ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*. Association for Computing Machinery, 2017, pp. 697–706. DOI: 10.1145/3097983.3098099.

[3] Saurabh Agrawal, Michael Steinbach, Daniel Boley, Snigdhansu Chatterjee, Gowtham Atluri, Anh The Dang, Stefan Liess and Vipin Kumar. "Mining Novel Multivariate Relationships in Time Series Data Using Correlation Networks". In: *IEEE Transactions on Knowledge and Data Engineering* 32.9 (2020), pp. 1798–1811. DOI: 10.1109/TKDE.2019.2911681.

[4] Sarah Aliko, Jiawen Huang, Florin Gheorghiu, Stefanie Meliss and Jeremy I Skipper. *"Naturalistic Neuroimaging Database"*. OpenNeuro, 2021. DOI: 10.18112/openneuro.ds002837.v2.0.0.

[5] Margaret L. Andersen and Lori Olsen. "Corporate Social and Financial Performance: A Canonical Correlation Analysis". In: *Academy of Accounting and Financial Studies Journal* 15 (2011), p. 17.

[6] David Arthur and Sergei Vassilvitskii. "K-Means++: The Advantages of Careful Seeding". In: *Proceedings of the Eighteenth Annual ACM-SIAM Symposium on Discrete Algorithms*. SODA '07. Society for Industrial and Applied Mathematics, 2007, pp. 1027–1035. DOI: 10.5555/1283383.1283494.

[7] Bharat Biswal, Zerrin F Yetkin, Victor M Haughton and James S Hyde. "Functional connectivity in the motor cortex of resting human brain using echo-planar MRI". In: *Magnetic resonance in medicine* 34.4 (1995), pp. 537–541. DOI: 10.1002/mrm.1910340409.

[8] "Canonical Correlation Analysis". In: *Applied Multivariate Statistical Analysis*. Berlin, Heidelberg: Springer Berlin Heidelberg, 2007, pp. 321–330. DOI: 10.1007/978-3-540-72244-1_14.

[9] Örjan Carlborg and Chris S. Haley. "Epistasis: too often neglected in complex trait studies?" In: *Nature Reviews Genetics* 5.8 (Aug. 2004), pp. 618–625. DOI: 10.1038/nrg1407.

[10] Chun-Hung Cheng, Ada Waichee Fu and Yi Zhang. "Entropy-based subspace clustering for mining numerical data". In: *Proceedings of the fifth ACM SIGKDD international conference on Knowledge discovery and data mining*. Association for Computing Machinery, 1999, pp. 84–93. DOI: 10.1145/312129.312199.

[11] Roger H.L. Chiang, Chua Eng Huang Cecil and Ee-Peng Lim. "Linear correlation discovery in databases: a data mining approach". In: *Data & Knowledge Engineering* 53.3 (2005), pp. 311–337. DOI: 10.1016/j.datak.2004.09.002.

[12] CoinGecko. *CoinGecko API*. https://www.coingecko.com/en/api. Accessed: 2021-07-13. 2021.

[13] Thomas M. Cover and Joy A. Thomas. *Elements of Information Theory*. USA: Wiley-Interscience, 2006.

[14] Abhimanyu Das and David Kempe. "Algorithms for Subset Selection in Linear Regression". In: *Proceedings of the Fortieth Annual ACM Symposium on Theory of Computing*. STOC '08. Association for Computing Machinery, 2008, pp. 45–54. DOI: 10.1145/1374376.1374384.

[15] Mayur Datar, Nicole Immorlica, Piotr Indyk and Vahab S. Mirrokni. "Locality-Sensitive Hashing Scheme Based on p-Stable Distributions". In: *Proceedings of the Twentieth Annual Symposium on Computational Geometry*. SCG '04. Association for Computing Machinery, 2004, pp. 253–262. DOI: 10.1145/997817.997857.

[16] Ronald Fagin, Amnon Lotem and Moni Naor. "Optimal Aggregation Algorithms for Middleware". In: *Journal of Computer and System Sciences* 66 (2001), pp. 614–656.

[17] Apache Software Foundation. *Apache Spark*. https://spark.apache.org/. Accessed: 2021-08-10. 2021.

[18] Simons Foundation. *SPARK for Autism*. https://sparkforautism.org/portal/page/autism-research/. Accessed: 2021-08-10. 2021.

[19] Simons Foundation. *SPARK Gene list*. https://d2dxtcm9g2oro2.cloudfront.net/wp-content/uploads/2020/07/13153839/SPARK_gene_list_July2020.pdf. Accessed: 2021-08-10. 2021.

[20] Kenji Fukumizu, Francis R. Bach and Arthur Gretton. "Statistical Consistency of Kernel Canonical Correlation Analysis". In: *Journal of Machine Learning Research* 8.14 (2007), pp. 361–383. URL: http://jmlr.org/papers/v8/fukumizu07a.html.

[21] Antonin Guttman. "R Trees: A Dynamic Index Structure for Spatial Searching". In: vol. 14. Jan. 1984, pp. 47–57. DOI: 10.1145/971697.602266.

[22] Joseph F Hair, Rolph E Anderson, Ronald Tatham and William C Black. *Multivariate data analysis with readings*. Macmillan Pub., 1984.

[23] Daniel A. Handwerker, Vinai Roopchansingh, Javier Gonzalez-Castillo and Peter A. Bandettini. "Periodic changes in fMRI connectivity". In: *NeuroImage* 63.3 (2012), pp. 1712–1719. DOI: 10.1016/j.neuroimage.2012.06.078.

[24] Wolfgang Karl Härdle. "Canonical Correlation Analysis". In: *Applied Multivariate Statistical Analysis*. Berlin, Heidelberg: Springer Berlin Heidelberg, 2007, pp. 321–330. DOI: 10.1007/978-3-540-72244-1_14.

[25] David R Hardoon and John Shawe-Taylor. "Sparse canonical correlation analysis". In: *Machine Learning* 83.3 (2011), pp. 331–353. DOI: 10.1007/s10994-010-5222-7.

[26] H. Hasan Örkcü. "Subset selection in multiple linear regression models: A hybrid of genetic and simulated annealing algorithms". In: *Applied Mathematics and Computation* 219.23 (2013), pp. 11018–11028. DOI: 10.1016/j.amc.2013.05.016.

[27] Trevor Hastie, Robert Tibshirani and Jerome Friedman. *The elements of statistical learning: data mining, inference, and prediction*. 2nd ed. Springer New York, 2009.

[28] Stephan Heunis, Rolf Lamerichs, Svitlana Zinger, Cesar Caballero-Gaudes, Jacobus F.A. Jansen, Bert Aldenkamp and Marcel Breeuwer. "Quality and denoising in real-time functional magnetic resonance imaging neurofeedback: A methods review". In: *Human Brain Mapping* 41.12 (2020), pp. 3439–3467. DOI: 10.1002/hbm.25010.

[29] Joe Hicklin, Cleve Moler, Peter Webb, Ronald F. Boisvert, Bruce Miller, Roldan Pozo and Karin Remington. *JAMA : A Java Matrix Package*. https://math.nist.gov/javanumerics/jama/. Accessed: 2021-8-24. 2021.

[30] Eamonn Keogh, Kaushik Chakrabarti, Michael Pazzani and Sharad Mehrotra. "Dimensionality reduction for fast similarity search in large time series databases". In: *Knowledge and information Systems* 3.3 (2001), pp. 263–286. DOI: 10.1007/PL00011669.

[31] Robert Kistler et al. "The NCEP/NCAR 50-year reanalysis: monthly means CD-ROM and documentation". In: *Bulletin of the American Meteorological Society* 82 (Feb. 2001), pp. 247–268. DOI: 10.1175/1520-0477(2001)082<0247:TNNYRM>2.3.CO;2.

[32] L. Kolotilina. "A generalization of Weyl's inequalities with implications". In: *Journal of Mathematical Sciences* 101 (Sept. 2000), pp. 3255–3260. DOI: 10.1007/BF02672770.

[33] Silvan Licher, Shahzad Ahmad, Hata Karamujić-Čomić, Trudy Voortman, Maarten J. G. Leening, M. Arfan Ikram and M. Kamran Ikram. "Genetic predisposition, modifiable-risk-factor profile and long-term dementia risk in the general population". In: *Nature Medicine* 25.9 (Aug. 2019), pp. 1364–1369. DOI: 10.1038/s41591-019-0547-7.

[34] Stefan Liess, Saurabh Agrawal, Snigdhansu Chatterjee and Vipin Kumar. "A Teleconnection between the West Siberian Plain and the ENSO Region". In: *Journal of Climate* 30.1 (2017), pp. 301–315. DOI: 10.1175/JCLI-D-15-0884.1.

[35] Jessica Lin, Eamonn Keogh, Stefano Lonardi and Bill Chiu. "A Symbolic Representation of Time Series, with Implications for Streaming Algorithms". In: *Proceedings of the 8th ACM SIGMOD Workshop on Research Issues in Data Mining and Knowledge Discovery*. DMKD '03. Association for Computing Machinery, 2003, pp. 2–11. DOI: 10.1145/882082.882086.

[36] Myles E. Mangram. "A Simplified Perspective of the Markowitz Portfolio Theory". In: *Global Journal of Business Research* 7.1 (2013), pp. 59–70. URL: https://ssrn.com/abstract=2147880.

[37] James Manyika, Michael Chui, Brad Brown, Jacques Bughin, Richard Dobbs, Charles Roxburgh and Angela Hung Byers. *Big data: The next frontier for innovation, competition, and productivity*. May 2011. URL: https://www.mckinsey.com/business-functions/mckinsey-digital/our-insights/big-data-the-next-frontier-for-innovation.

[38] Fukuda Megumi, Ayumu Yamashita, Mitsuo Kawato and Hiroshi Imamizu. "Functional MRI neurofeedback training on connectivity between two regions induces long-lasting changes in intrinsic functional network". In: *Frontiers in Human Neuroscience* 9.MAR (2015). DOI: 10.3389/fnhum.2015.00160.

[39] Koen Minartz, Jens d'Hondt and Odysseas Papapetrou. *Multivariate correlations discovery in static and streaming data*. Tech. rep. Available in https://github.com/JdHondt/CorrelationDetective. Eindhoven University of Technology, 2021.

[40] Ileena Mitra, Alinoë Lavillaureix, Erika Yeh, Michela Traglia, Kathryn Tsang, Carrie E. Bearden, Katherine A. Rauen and Lauren A. Weiss. "Reverse Pathway Genetic Approach Identifies Epistasis in Autism Spectrum Disorders". In: *PLOS Genetics* 13.1 (Jan. 2017), pp. 1–27. DOI: 10.1371/journal.pgen.1006516.

[41] Sandy Moens, Emin Aksehirli and Bart Goethals. "Frequent itemset mining for big data". In: *2013 IEEE International Conference on Big Data*. IEEE. 2013, pp. 111–118. DOI: 10.1109/BigData.2013.6691742.

[42] Douglas Montgomery. *Applied statistics and probability for engineers*. Hoboken, NJ: Wiley, 2011.

[43] Abdullah Mueen, Suman Nath and Jie Liu. "Fast Approximate Correlation for Massive Time-Series Data". In: *Proceedings of the 2010 ACM SIGMOD International Conference on Management of Data*. SIGMOD '10. Association for Computing Machinery, 2010, pp. 171–182. DOI: 10.1145/1807167.1807188.

[44] Emmanuel Müller, Ira Assent, Ralph Krieger, Stephan Günnemann and Thomas Seidl. "DensEst: Density Estimation for Data Mining in High Dimensional Spaces". In: *Proceedings of the Ninth SIAM International Conference on Data Mining*. Apr. 2009, pp. 173–184. DOI: 10.1137/1.9781611972795.16.

[45] Hoang Vu Nguyen, Emmanuel Müller, Periklis Andritsos and Klemens Böhm. "Detecting Correlated Columns in Relational Databases with Mixed Data Types". In: *Proceedings of the 26th International Conference on Scientific and Statistical Database Management*. SSDBM '14. Association for Computing Machinery, 2014. DOI: 10.1145/2618243.2618251.

[46] Hoang Vu Nguyen, Emmanuel Müller, Jilles Vreeken, Pavel Efros and Klemens Böhm. "Multivariate maximal correlation analysis". In: *ICML* 3 (Jan. 2014), pp. 775–783.

[47] Philon Nguyen and Nematollaah Shiri. "Fast Correlation Analysis on Time Series Datasets". In: *Proceedings of the 17th ACM Conference on Information and Knowledge Management*. CIKM '08. Association for Computing Machinery, 2008, pp. 787–796. DOI: 10.1145/1458082.1458187.

[48] National Oceanic and Atmospheric Administration. *NOAA Integrated Surface Dataset (Global)*. https://www.ncei.noaa.gov/access/search/dataset-search. Accessed: 2021-07-30. 2021.

[49] Oracle. *Java 8*. https://www.java.com/en/download/help/java8.html. Accessed: 2021-8-24. 2021.

[50] Lance Parsons, Ehtesham Haque and Huan Liu. "Subspace Clustering for High Dimensional Data: A Review". In: *SIGKDD Explor. Newsl.* 6.1 (June 2004), pp. 90–105. DOI: 10.1145/1007730.1007731.

[51] Andy Patrizio. *IDC: Expect 175 zettabytes of data worldwide by 2025*. Oct. 2018. URL: https://www.networkworld.com/article/3325397/idc-expect-175-zettabytes-of-data-worldwide-by-2025.html.

[52] Camilo Rostoker, Alan Wagner and Holger Hoos. "A Parallel Workflow for Real-time Correlation and Clustering of High-Frequency Stock Market Data". In: *2007 IEEE International Parallel and Distributed Processing Symposium*. 2007, pp. 1–10. DOI: 10.1109/IPDPS.2007.370216.

[53] Venu Satuluri and Srinivasan Parthasarathy. "Bayesian Locality Sensitive Hashing for Fast Similarity Search". In: *Proceedings of the VLDB Endowment* 5.5 (Jan. 2012), pp. 430–441. DOI: 10.14778/2140436.2140440.

[54] Robert Tibshirani. "Regression Shrinkage and Selection via the Lasso". In: *Journal of the Royal Statistical Society. Series B (Methodological)* 58.1 (1996), pp. 267–288. URL: http://www.jstor.org/stable/2346178.

[55] VLDB. *VLDB 2022 - 48th International Conference on Very Large Data Bases*. https://vldb.org/2022/. Accessed: 2021-08-11. 2021.

[56] Hao-Ting Wang, Jonathan Smallwood, Janaina Mourao-Miranda, Cedric Huchuan Xia, Theodore D. Satterthwaite, Danielle S. Bassett and Danilo Bzdok. "Finding the needle in a high-dimensional haystack: Canonical correlation analysis for neuroscientists". In: *NeuroImage* 216 (2020), p. 116745. DOI: 10.1016/j.neuroimage.2020.116745.

[57] Jianji Wang, Nanning Zheng, Badong Chen, Pei Chen, Shitao Chen, Ziyi Liu, Fei-Yue Wang and Bao Xi. "Multivariate Correlation Entropy and Law Discovery in Large Data Sets". In: *IEEE Intelligent Systems* 33.5 (Sept. 2018), pp. 47–54. DOI: 10.1109/MIS.2018.2877282.

[58] Satosi Watanabe. "Information Theoretical Analysis of Multivariate Correlation". In: *IBM Journal of Research and Development* 4.1 (1960), pp. 66–82. DOI: 10.1147/rd.41.0066.

[59] Yingjun Wu, Jia Yu, Yuanyuan Tian, Richard Sidle and Ronald Barber. "Designing Succinct Secondary Indexing Mechanism by Exploiting Column Correlations". In: *Proceedings of the 2019 International Conference on Management of Data*. SIGMOD '19. Association for Computing Machinery, 2019, pp. 1223–1240. DOI: 10.1145/3299869.3319861.

[60] Xiang Zhang, Feng Pan, Wei Wang and Andrew Nobel. "Mining non-redundant high order correlations in binary data". In: *Proceedings of the VLDB Endowment* 1.1 (Aug. 2008), pp. 1178–1188. DOI: 10.14778/1453856.1453981.

[61] Yunyue Zhu and Dennis Shasha. "StatStream: Statistical Monitoring of Thousands of Data Streams in Real Time". In: *Proceedings of the 28th International Conference on Very Large Data Bases*. VLDB '02. VLDB Endowment, 2002, pp. 358–369. DOI: 10.5555/1287369.1287401.

[62] Anna Zilverstand, Bettina Sorger, Jan Zimmermann, Amanda Kaas and Rainer Goebel. "Windowed Correlation: A Suitable Tool for Providing Dynamic fMRI-Based Functional Connectivity Neurofeedback on Task Difficulty". In: *PLOS ONE* 9.1 (2014), pp. 1–13. DOI: 10.1371/journal.pone.0085929.

Appendix A

Additional Experimental Results

Table A.1 shows the results of exploratory experiments where the clustering parameters are varied. Notice that the column $1 - \frac{1}{2d}\epsilon_{\text{start}}^2$ is the value of ρ that corresponds to the value of ϵ_{start}. Observe that lowering K, as well as increasing α generally helps performance a bit; both lowering K and increasing α limit the speed with which CD Algorithm 2 reduces the radii and number of points in clusters. This indicates that it is better to gradually reduce the sizes of each clusters, in order to try to prune large cluster combinations, as opposed to quickly reducing their sizes to spend less times on internal nodes in the cluster combination tree (as illustrated in Figure 4.4b).

However, the effects of the clustering parameters are not too strong, and the values of the query parameters (δ, k) have a much stronger effect on running time. This is also confirmed by Table A.2, which shows the Pearson correlation of each parameter's value with the running time. The effect of the clustering parameters is weaker than the effect of the query parameters, which in turn is again weaker than the correlation of runtime with output statistics like the size of the result set and the final value of the rolling τ, τ_k. As such, we conclude that the effect of the clustering parameters is not negligible. Nevertheless, given the already huge parameter space of CD, it is impossible to find optimal clustering parameters for each query configuration within the timespan of this thesis; as a rule of thumb setting K reasonably low (say 10 - 30) and α reasonably high (≥ 0.8) will result in performance that is sufficiently optimized for our purpose.

Dataset	Correlation Pattern	τ	δ	$1-\frac{1}{2d}\epsilon^2_{\text{start}}$	α	K (K-means)	k (top-k)	running time (s)	$\lvert\mathcal{R}\rvert$	τ_k
Sustainable stocks	**mc**$(1,3)$	0.6	0.001	0.25	0.65	100	50	174	50	0.953
							500	978	500	0.942
						25	50	135	50	0.953
							500	593	500	0.942
					0.825	100	50	153	50	0.953
							500	594	500	0.942
						25	50	159	50	0.953
							500	693	500	0.942
				0.5	0.65	100	50	296	50	0.953
							500	910	500	0.942
						25	50	134	50	0.953
							500	576	500	0.942
					0.825	100	50	136	50	0.953
							500	529	500	0.942
						25	50	142	50	0.953
							500	648	500	0.942
			None	0.25	0.65	100	50	184	50	0.977
							500	1419	500	0.967
						25	50	188	50	0.977
							500	829	500	0.967
					0.825	100	50	225	50	0.977
							500	893	500	0.967
						25	50	175	50	0.977
							500	955	500	0.967
				0.5	0.65	100	50	387	50	0.977
							500	1223	500	0.967
						25	50	188	50	0.977
							500	836	500	0.967
					0.825	100	50	161	50	0.977
							500	756	500	0.967
						25	50	178	50	0.977
							500	927	500	0.967
SLP	**mp**(4)	0.6	0.15	0.25	0.65	100	100	250	100	0.864
							∞	667	1843	0.6
						25	100	231	100	0.864
							∞	611	1843	0.6
					0.825	100	100	238	100	0.864
							∞	638	1843	0.6
						25	100	235	100	0.864
							∞	623	1843	0.6
				0.5	0.65	100	100	233	100	0.864
							∞	628	1843	0.6
						25	100	235	100	0.864
							∞	606	1843	0.6
					0.825	100	100	219	100	0.864
							∞	579	1843	0.6
						25	100	225	100	0.864
							∞	584	1843	0.6
			None	0.25	0.65	100	100	164	100	0.989
							1000	166	1000	0.982
						25	100	150	100	0.989
							1000	168	1000	0.982
					0.825	100	100	158	100	0.989
							1000	166	1000	0.982
						25	100	151	100	0.989
							1000	162	1000	0.982
				0.5	0.65	100	100	157	100	0.989
							1000	173	1000	0.982
						25	100	144	100	0.989
							1000	160	1000	0.982
					0.825	100	100	161	100	0.989
							1000	168	1000	0.982
						25	100	153	100	0.989
							1000	168	1000	0.982

Table A.1: Exploratory experiments for tuning the clustering parameters

	δ	$1-\frac{1}{2d}\epsilon^2_{\text{start}}$	α	K	Top-k	$\lvert\mathcal{R}\rvert$	τ_k
Running time	0.18	-0.03	-0.08	0.05	0.2	0.47	-0.44

Table A.2: Correlation values of Running times with different parameters