Imperial College London

Department of Mathematics

From Cross-Sectional to Network Momentum: Enhancing Systematic Trend-Following Strategies

Author: Linze Li (CID: 01869053)

A thesis submitted for the degree of

MSc in Mathematics and Finance, 2023-2024

Declaration

The work contained in this thesis is my own work unless otherwise stated.

Acknowledgements

I would like to thank the systematic strategies desk at Ocean Leonid for the opportunity to undertake this project. I am particularly grateful to Dr William Ferreira, Leonardo Marroni, Irene Perdomo and Lorenzo Reati for their guidance in developing my understanding of systematic trading.

I extend my gratitude to Dr Yonatan Shadmi and Dr Anthony Coache for their continuous support and valuable feedback during our weekly meetings, which have been instrumental in writing this thesis.

I remain forever grateful to my late grandmother, whose enduring influence continues to inspire me every day.

Abstract

In financial markets, it is often observed that some assets lead in price trends while others follow with a time delay, creating what is known as cross-sectional momentum spillover or the 'lead-lag' relationship between paired markets. In this thesis, we introduce a method for transforming cross-sectional momentum into network momentum by quantifying the interconnectedness of market leadership and individual momentum. We develop a systematic trend-following strategy based on this network momentum. Our framework incorporates two state-of-the-art lead-lag detection methods and a graph learning model to generate a new trading signal for momentum strategies. We validate our framework using daily price data from 28 real-world futures spanning four asset classes from 2005 to 2024, alongside 100 sets of synthetic price data. Our results demonstrate that the network momentum strategy significantly improves the Sharpe ratio by 29% on average compared to the individual momentum strategy. It also effectively reduces losses in short positions by 35% and increases the skewness of monthly returns by 29%. This thesis pioneers research into transforming cross-sectional lead-lag relationships into network momentum spillover. The proposed framework demonstrates broad generalisability and consistent effectiveness across various market combinations and is independent of historical market trends.

Contents

List of Figures

List of Tables

Introduction

Trend-following strategies have existed as an investment style for a very long time. They are widely adopted by commodity trading advisors, over-the-counter markets, macro hedge funds, and both sell-side and buy-side practitioners. These strategies share a common premise: market returns exhibit auto-correlation, with winners likely to continue performing well and losers continuing to underperform. Consequently, practitioners typically take long positions in markets with recent positive performance and short positions in those generating negative returns. Therefore, these are also known as momentum strategies. In this manuscript, we propose a trend-following strategy that trades markets based not only on their recent performance but also on the performance of other interconnected markets. This strategy uses state-of-the-art lead-lag detection and graph learning models to identify and exploit these inter-market influences.

The time series momentum strategy is arguably the most fundamental trend-following strategy, where portfolios are constructed based on the momentum of individual markets, meaning positions are established solely according to recent market performance. The persistence of market returns has been extensively studied from macroeconomic perspectives. For instance, industrial growth rates have a significant impact on momentum profits [41]; investor behaviours, such as delayed information reception and asynchronous response timings, support the slow information diffusion hypothesis [27, 28]. Behavioural biases like conservatism may also encourage premature selling or prolonged holding of assets [5]. Such persistence in market returns continues until significant deviations from price fundamentals eventually trigger a market reversal [66]. The profitability of time series momentum strategies is demonstrated across various markets, showing that purchasing stocks that perform well in recent months and selling those that show poor returns results in higher profits [30, 58, 16]. This profitability has been rigorously validated through statistical experiments to confirm it is not due to random chance [31, 29]. A popular method for measuring time series momentum is the crossover of moving averages of recent prices [39, 7, 45]. The underlying premise is that if the short-term average price crosses above the long-term average price from below, it indicates an expected increase in price, suggesting a potential upward trend and, conversely, a downward trend if it crosses below.

The momentum strategy extends beyond individual markets. [22] observes that high equity returns in one year can predict high corporate bond returns the following year despite bonds lacking inherent momentum. This 'cross-sectional momentum spillover' is primarily due to the bond market's delayed reaction to equity performance, known as the 'lead-lag effect'. Previous literature has identified multiple drivers for the lead-lag effect, including factors such as company size [43], institutional ownership levels [3], analyst coverage [11], and industry affiliation [48, 23]. Numerous studies have explored systematic approaches to capturing the lead-lag effect. For instance, [12] utilised the difference in the cross-correlation function based on Pearson correlation, while [69] employed the signed normalised area under the curve of the cross-correlation function as an indicator. Further, [8] experimented with alternatives to Pearson correlation, such as Kendall rank correlation [35], distance correlation [62], and mutual information from discretised time series values [21]. We refer interested readers to [10, 67, 44] for additional methods for detecting lead-lag

effects.

After computing lead-lag metrics pairwisely, many studies suggest employing ranking algorithms to identify assets most likely to lead or follow [13, 8, 1]. Positions for the followers are established based on the average of the leaders' performance. For example, if leaders exhibit a negative average return, followers are shorted in anticipation of a similar downward trend. To rank markets based on their leadership influence cross-sectionally, the literature employs methods such as RowSum Rank [13, 72, 8], PageRank [69, 6], and machine learning approaches like the Learning-to-rank algorithm [63, 52]. Another strategy involves clustering lead-lag metrics using algorithms such as clustering by industry sectors [28, 10], k-means [72], and spectral and Hermitian clustering [8, 13]. This data-driven methodology, prevalent in existing studies [6, 69], suggests establishing positions within lagging clusters based on the average performance of markets in the leading cluster. This approach is used to capitalise on trend-following opportunities or to construct opposite positions to counteract mean-reverting behaviours [13].

To the best of our knowledge, quantitative research that measures the influence of leading markets on the performance of lagging markets is currently limited, especially when portfolios span multiple industries. It is pointed out in [56] that, although previous studies have identified momentum spillover across various sectors—such as equities and bonds [22, 26], equities and foreign currencies [17], currency news and bonds in emerging markets [71], and between crude oil indexes and equities [20]—the absence of firm-like economic and fundamental linkages in commodities markets complicates the identification of connections, such as those between orange juice and natural gas.

Moreover, while existing studies predominantly focus on statistically examining the momentum spillover effect and use it as a market selection mechanism for trading followers in exchanges, they fail to quantify and aggregate this momentum spillover into a new trading signal for portfolio construction. To bridge this gap, the existing literature suggests leveraging network theory. For instance, [70] uses edge centrality to quantify the importance of supplier-customer relationships, and [56] explores the 'network momentum' spillover across industries, aggregating momentum from commodities, equities, bonds, and currencies to create a novel trading signal. Specifically, the latter research employs ideas from graph learning, treating each market as a node within a graph. This approach uses time series momentum features, including moving average crossover signals and exponentially weighted returns, as signal processes for each node. They then solve a convex optimisation problem to approximate the weighted graph adjacency matrix. The edges of this graph elucidate the complex relationships across markets, with the magnitude of each edge reflecting the strength of similarity in momentum features between market pairs. Following this model, the time series momentum of other markets is weighted and used in a linear regression to predict the next day's returns. Subsequently, a portfolio is constructed by assigning binary positions—either 1 or -1—based on the sign of the predicted future returns, reflecting long or short positions in alignment with existing literature [72, 63, 52]. However, such binary betting on positions based on momentum direction may not be optimal because this approach could lead to a discontinuous model, losing both convexity and positive skewness in returns [45]. Similarly, [56] notes that their strategy may not adequately address risk characteristics, potentially increasing exposure to significant downside movements.

Our objective is to transform cross-sectional momentum spillover—essentially the pairwise lead-lag effect—into the concept of 'network momentum' spillover and construct a systematic trend-following trading strategy that preserves the positive skewness of returns. We achieve this in three steps. First, we employ two non-parametric methods to identify non-linear and non-synchronous lead-lag relationships between market pairs. Second, using graph learning models, we quantify the interconnected lead-lag strengths and ag-

gregate the time series momentum of each market, incorporating weighted contributions from connected markets into a novel trading signal. Third, we construct a portfolio by converting this trading signal into a position signal, ensuring a positively skewed return can consistently be achieved.

In the first step, we employ two state-of-the-art models to identify the lead-lag relationship and quantify the leadership between pairs of assets. The first model, based on the Lévy area of pairwise market returns as proposed by [13], effectively identifies both linear and nonlinear relationships at a fixed lag, such as the one-day lag effect. The second model utilises the dynamic time warping algorithm on pairwise market returns, in line with established literature [72, 65]. This model relaxes the fixed lag assumption and adeptly handles non-synchronised time series of varying lengths. Building upon these foundations, we explore various advanced dynamic time warping algorithms to capture the co-movement between two markets' returns. We then calculate pairwise lead-lag scores for all combinations of market pairs and use them to construct a skew-symmetric matrix known as the lead-lag matrix.

In the second step, we apply the graph learning model proposed by [34], which has also been utilised by [56] for analysing time series momentum features and approximating the graph adjacency matrix through convex optimisation. We propose to apply this model to our lead-lag matrix, treating each market as a node where the leadingness to other markets represents a signal process. This approach effectively analyses the relationships between lead-lag effects, turning the cross-sectional momentum into network momentum and highlighting the strength of the relationship via the non-negative edge values in the adjacency matrix.

Once the network adjacency matrix is established, we apply it to the time series momentum features to aggregate the momentum from all connected markets, thereby creating a spillover effect where the momentum of each market is influenced by its interconnected markets. The aggregated momentum signal then serves as input to a response function [45], which determines the position needed to follow the observed trend. We assess the effectiveness of our approach by comparing the profitability of our approach on bootstrapped datasets against a strategy that relies solely on time series momentum.

In contrast to the extant literature in [13, 56], where the profits and losses (PnL) for day $t+1$ are calculated by multiplying the market return between days t and $t+1$ with the position signal from day t — a position generated using close price up to and including day t, our approach addresses a key limitation. This conventional approach assumes that trades can be executed immediately upon receiving the closing price information on the same day, which is usually not feasible in real-life trading scenarios. We take a more conservative approach by introducing an additional lag in our model: a signal generated using price information up to day t establishes a trade to be completed by the end of day $t+1$, which then accrues PnL based on the price change from day $t+1$ to day $t+2$. Transaction costs, calculated independently of the price, are estimated as half of the bid-ask spread at the closing price and are accrued when entering the position on day $t+1$.

We backtest the proposed network momentum strategy on 28 futures contracts across four industry sectors—energy, agriculture, metals, and equities—covering an out-of-sample period from 2005 to 2024. This strategy outperforms the traditional time series momentum strategy, demonstrating a 29% increase in the Sharpe ratio, a 19% reduction in transaction costs, and a 29% improvement in positive skewness of returns on bootstrapped datasets. Notably, our strategy also reduces losses in short positions by 35% compared to the traditional approach on average on 100 sets of bootstrapped price data. These results highlight the strong generalisability of our model across various markets and its consistent superiority over the classical time series momentum strategy.

The contributions of this paper are threefold. First, it expands on existing literature by employing a range of dynamic time warping algorithms for lead-lag detection, moving beyond the classical approach typically used in prior studies. Secondly, our study pioneers the treatment of the lead-lag matrix as a graph signal, transforming cross-sectional momentum into network momentum. Thirdly, to the best of our knowledge, this is the first study to demonstrate the effectiveness of lead-lag detection and network momentum spillover in systematic trend-following strategies within a realistic trading environment through rigorous backtesting and empirical testing on multiple bootstrapped datasets.

The remainder of the paper is organised as follows. Chapters 1 and 2 present the mathematical frameworks used to identify the lead-lag relationship. Chapter 3 details the graph learning model employed to capture network momentum. Chapter 4 explains the strategy setup and subsequent performance analysis. We conclude the paper with the final thoughts and future directions and provide proofs, auxiliary data, motivation and methodology for bootstrapping in Appendices A, B and C.

Chapter 1

Signature Based Lévy Area

In this chapter, we introduce the first of our two models to identify pairwise non-linear lead-lag relationships with a fixed temporal gap ℓ, where ℓ is a positive integer representing the number of time units. For a return series of market n at each time $t = 1, 2, \ldots, T$, denoted as $R_{t,n}$, we investigate whether there exists a leading market m such that its return at time $t - \ell$, denoted as $R_{t-\ell,m}$, has predictive power over $R_{t,n}$. We demonstrate that the Lévy area between the series $R_{t,m}$ and $R_{t,n}$ can provide insights into the leading behaviour of market m relative to market n. From this analysis, we construct a lead-lag matrix to serve as an input for our graph learning model.

Preliminary knowledge regarding the signature method is outlined in Section 1.1, with its application to detecting lead-lag relationships discussed in Section 1.2. We summarise our algorithm for constructing the lead-lag matrix in Section 1.3.

1.1 Path and Signature

In this section, we mostly follow the manuscript in [15], introducing some fundamental definitions and knowledge of a path and signature.

Definition 1.1.1 (Path)**.** A path $X \in \mathbb{R}^d$ is defined as a continuous mapping from an interval $[a, b]$ to $\mathbb{R}^d$, that is

$$X : [a, b] \to \mathbb{R}^d.$$

Following the convention in [15], we use $X_t = X(t)$ to denote the dependence on the parameter $t \in [a, b]$.

Definition 1.1.2 (Coordinate path)**.** For a path $X : [a, b] \to \mathbb{R}^d$, the coordinate path is defined as

$$(X_t^1, \ldots, X_t^d),$$

where each X^i is a real-valued path from $[a, b]$ to $\mathbb{R}$.

For any single index $i \in \{1, \ldots, d\}$, we define

$$S(X)_{a,t}^i := \int_{a<s<t} dX_s^i = X_t^i - X_0^i, \tag{1.1.1}$$

which is the increment of the $i-$th coordinate of the path at time $t \in [a, b]$. For any pair $i, j \in \{1, \ldots, d\}$, the double-iterated integral is defined as

$$S(X)_{a,t}^{i,j} := \int_{a<s<t} S(X)_{a,s}^i dX_s^j = \int_{a<r<s<t} dX_r^i dX_s^j, \tag{1.1.2}$$

where $S(X)^i_{a,s}$ is the coordinate increment in (1.1.1). Notice that for $i = j \in [1, \dots, d]$,

$$S(X)^{i,i}_{a,b} = \frac{(S(X)^i_{a,b})^2}{2} = \frac{(X^i_b - X^i_a)^2}{2}.$$

We can further extend this definition to a $k-$fold iterated integral of path X as

$$S(X)^{i_1,\dots,i_k}_{a,t} := \int_{a<s<t} S(X)^{i_1,\dots,i_{k-1}}_{a,s} dX^{i_k}_s = \int_{a<t_k<t} \cdots \int_{a<t_1<t_2} dX^{i_1}_{t_1} \cdots dX^{i_k}_{t_k}.$$

Following the convention in [15], we assume, unless stated otherwise, that paths are piecewise differentiable and smooth to ensure that the iterated integrals of the path existed as Riemann-Stieltjes integrals and the path has derivatives of all orders.

Definition 1.1.3 (Signature)**.** [15, Definition 1, page 5] The signature of a path $X : [a, b] \to \mathbb{R}^d$, denoted by $S(X)_{a,b}$, is the collection (infinite series) of all the iterated integrals of X. Formally, $S(X)_{a,b}$ is the sequence of real numbers

$$S(X)_{a,b} := (1, S(X)^1_{a,b}, \cdots, S(X)^d_{a,b}, S(X)^{1,1}_{a,b}, S(X)^{1,2}_{a,b}, \cdots),$$

where the 'zeroth' term, by convention, is equal to 1, and the superscripts run along the set of all multi-indexes

$$W = \{(i_1, \dots, i_k) | k \geq 1, i_1, \dots, i_k \in \{1, \dots, d\}\}.$$

The $k-$th level of the signature is the finite collection of all terms $S(X)^{i_1,\dots,i_k}_{a,b}$ where the multi-index is of length k [15]. For example, the first level is

$$\{S(X)^1_{a,b}, \dots, S(X)^d_{a,b}\},$$

and the second level is

$$\{S(X)^{1,1}_{a,b}, \dots, S(X)^{1,d}_{a,b}, S(X)^{2,1}_{a,b}, \dots, S(X)^{d,d}_{a,b}\}.$$

We hereby discuss the geometric intuition of the first two levels. The first level is always an increment of the path X because of its Formula (1.1.1). For the geometric interpretation of the second level with terms $S(X)^{i,j}_{a,b}$ for $i \neq j$, one could consider the Lévy area.

Definition 1.1.4 (Lévy Area)**.** [15, Page 10] Given a two-dimensional path $\{X^1_t, X^2_t\}$, the Lévy area measures the signed area enclosed by the path and the chord connecting the endpoints over the interval $[a, b]$. Mathematically, it is defined as

$$A^{\text{Lévy}} := \frac{1}{2}\left(\int_{a<r<s<t} dX^1_r dX^2_s - \int_{a<r<s<t} dX^2_r dX^1_s\right) = \frac{1}{2}(S(X)^{1,2}_{a,b} - S(X)^{2,1}_{a,b}),$$

where $S(X)^{i,j}_{a,b}$ denotes the iterated integral (1.1.2) of the components X^i and X^j.

1.2 Application in Lead-Lag Detection

There has been extensive research on using the knowledge of signature to identify the lead-lag relationship between two series. As mentioned in [25], given a path in $\mathbb{R}^2$ defined by $X_t = \{X^1_t, X^2_t\}$ for $t \in [a, b]$, if an increase (resp. decrease) in the component X^1 is followed by an increase (resp. decrease) in the component X^2, then the Lévy area $A^{\text{Lévy}}$ in Definition 1.1.4, is positive. Conversely, if the movements of X^1 and X^2 are in opposite

directions, then $A^{\text{Lévy}}$ is negative. For discrete processes X_t^1 and X_t^2, the Lévy area can be expressed as:

$$A^{\text{Lévy}} = \frac{1}{2}\left(\sum_{a<s<b}(-X_s^i X_{s-1}^j + X_s^j X_{s-1}^i) + X_a^i(X_a^j - X_b^j) + X_a^j(X_b^i - X_a^i)\right). \tag{1.2.1}$$

We provide the derivation of this formula in Appendix A.1.

An example is shown in Figure 1.1, where the notation $A^{\text{Lévy}}_{[s,t]}$ represents the Lévy area between the processes X^1 and X^2 from the initial point s to the endpoint t along the direction of X^1. Initially, as X^1 increases from 0 to 1.46, X^2 follows this increase, resulting in a positive Lévy area, with $A^{\text{Lévy}}_{[0,1.46]} = 1.17$. The increase continues until X^1 reaches 2.48, at which point X^2 reaches its maximum value, corresponding to a Lévy area of $A^{\text{Lévy}}_{[0,2.48]} = 1.69$. Beyond this point, X^2 moves against X^1, causing the Lévy area to decrease, leading to a negative Lévy area of $A^{\text{Lévy}}_{[0,4.89]} = -0.99$. Moreover, considering only the Lévy area enclosed from the maximum to the minimum of X^2 where they only exhibit movement in the opposite direction, the area is $A^{\text{Lévy}}_{[2.48,5.62]} = -3.2$, which is significantly larger in magnitude.

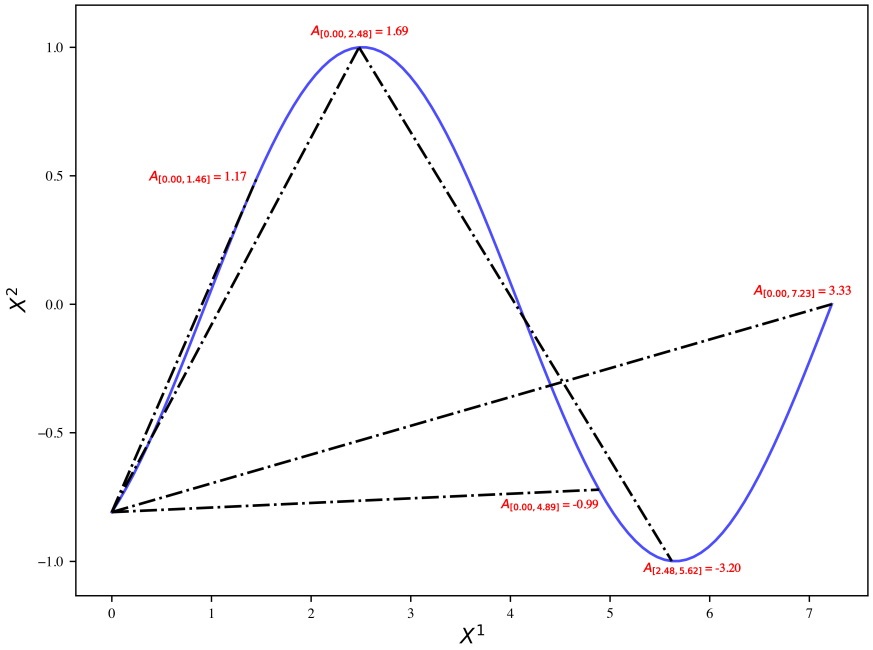


Figure 1.1: Example of the change in Lévy area enclosed between two discrete processes at different endpoints.

However, as pointed out in [8], this lead-lag metric cannot distinguish difference between:

1. $i \to j$ with negative association, that is X^j moves against X^i,
2. $i \leftarrow j$ with positive association, that is X^j leads X^i with same moving direction.

It is natural to apply this detection method to financial series. [13] shows that the Lévy area between the normalised returns of some markets at n and m, $R_{t,n}$ and $R_{t,m}$,

can indicate the lead-lag relationships for models of the form

$$R_{t,n} = \beta_\ell f(R_{t-\ell,m}) + \epsilon_t, \tag{1.2.2}$$

where f is any continuous function where ℓ is a positive integer with the same units as those of t. We present their result in the following theorem.

Theorem 1.2.1. *[13, Theorem 1, page 9] Assume $\{X_\tau^i\}_{\tau=s}^t$ and $\{X_\tau^j\}_{\tau=s}^t$ are two independent random processes with zero mean, unit variance, and symmetric distribution, and both satisfy* (1.2.2) *over a time interval [s,t]. Then, the sign of the Lévy area $A_{i,j}^{Lévy}$ between X_τ^i and X_τ^j is the same as the sign of ℓ if and only if $\ell = \pm 1$. In addition, if $\ell = \pm 1$ and the third derivative of the function f exists, there is a constant K such that for all pairs (i,j), we have*

$$E[A_{i,j}^{Lévy} - K\beta_\ell] = \frac{M}{6}\beta_\ell E[\sum_{s<a<t} f'''(\xi_{a-1}^j)(X_{a-1}^j)^4]$$

for some constant M and $|\xi_{a-1}^j| < |X_{a-1}^j|$.

Proof. See [13, Appendix C]. □

As [13] points out, this theorem suggests that the Lévy area only has predictability to the sign of ℓ when $\ell = \pm 1$, but by no means would it constrain us only to consider one day lag since one could always calculate the multi-step returns of a market to relax this constraint. [13] also mentions that the magnitude of the score quantifies the strength of the lead-lag relationship, which aligns with the result in Figure 1.1. However, the efficiency of its quantifying the strength depends on the behaviour of the function f, only when the third derivative of f is small over the unit disc, $A_{i,j}^{\text{Lévy}}$ is an accurate estimation of the lead-lag strength.

1.3 Computation of Lead-Lag Matrix Using Lévy Area

In this section, we outline our method for constructing the lead-lag metric matrix. Handling missing values in the price series is crucial since market returns are not always synchronised. For instance, on certain days, data from some markets may be missing, likely due to market closures. Consider the following series:

$$Y = [1, 3, \star, 5, 7, \star, \star, 9, 1, 4],$$

illustrated in Figure 1.2(a), where $\star$ indicates missing values. We use the rectilinear interpolation to transform the discrete series into a continuous path by adding auxiliary points marked in empty blue circles, which follows the convention in [15].

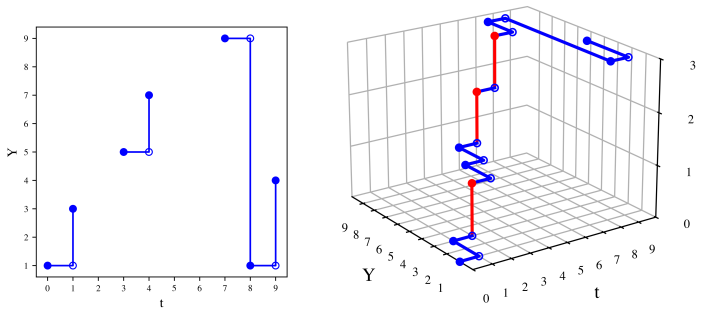
To address this, we define an indicator series R, where 1 represents a missing value and 0 an observed value:

$$t = [0, 1, 2, 3, 4, 5, 6, 7, 8, 9]$$
$$R = [0, 0, 1, 0, 0, 1, 1, 0, 0, 0].$$

According to [15], we embed the indicator vector R into the path Y by lifting the path into a higher dimension. Every time we have a missing point, we jump from the 'observed' to the 'missing' dimension and fill in the missing place with the same value as seen before. This is known as the 'feedforward' method since it extends the last observed value until a new observation is available. Then the filled observation is

$$\tilde{Y} = [1, 3, 3, 5, 7, 7, 7, 9, 1, 4].$$

We present the 3-dimensional path in Figure 1.2(b), where red lines and dots indicate extensions into a new dimension from the last observed data points. It is important to note that this feedforward approach should be applied directly to the price series. We can only accurately calculate the returns from the filled price series after addressing the gaps in the price data. This process effectively equates to inserting zeros into the return series for each market on days when data is missing.



(a) (b)

Figure 1.2: Illustration of a discrete path with missing values (left) and the same path in a higher dimension after interpolation to fill the gaps (right).

Based on the assumption in Theorem 1.2.1, we must standardise the returns of each market to zero mean and unit variance. This approach aligns with the convention in [8], where it is noted that the absolute value of this specific metric increases with the volatility of the underlying price series. The final step involves calculating the pairwise Lévy area between the markets using Formula (1.2.1). In [13], the authors assume that the return of assets at the start of the time series is 0, thereby reducing their formula to:

$$A = \frac{1}{2} \sum_{a<s<b} (-X_s^i X_{s-1}^j + X_s^j X_{s-1}^i).$$

It is important to note that our formula (1.2.1) does not change the results in Theorem 1.2.1. Moreover, in practice, when we truncate part of the historical data as training data to fit the model, we typically have the return of the market at the first point in our dataset. We summarise this algorithm in Table 1.1.

Table 1.1 Algorithm for using Lévy area for lead-lag metric computation

Require: Feature matrix $\mathbf{X_t} \in \mathbb{R}^{T \times M}$ representing market returns observed at time t, where T denotes the number of lookback days, and M represents the number of markets.

Ensure: Lead-lag metric matrix $\mathbf{V}_t \in \mathbb{R}^{M \times M}$

Fill in the missing values of each market by zeros.
Standardise each column to have zero mean and unit variance.
Initialise an empty matrix $\mathbf{V}_t$ with shape (M, M).
for $i \leftarrow 1$ to M **do**
    for $j \leftarrow i + 1$ to M **do**
        Take out paired returns series of market i and j, X^i and X^j.
        Let $a = t - T$ and $b = t$ be the start and end indices of the time series.
        Calculate the Lévy area enclosed by the two series and the chord connecting the endpoints according to (1.2.1):
        $A_{i,j} = \frac{1}{2}\left(\sum_{s=a+1}^{b}(-X_s^i X_{s-1}^j + X_s^j X_{s-1}^i) + X_a^i(X_a^j - X_b^j) + X_a^j(X_b^i - X_a^i)\right)$
        Set $\mathbf{V}_t[i][j] = A_{i,j}$
        Set $\mathbf{V}_t[j][i] = -A_{i,j}$
    end for
end for
return $\mathbf{V}_t$

Chapter 2

Dynamic Time Warping

In this chapter, we introduce our second model, which employs dynamic time warping to identify lead-lag relationships. Compared to the signature-based Lévy area model proposed in Chapter 1, this model differs in two major respects: firstly, it does not assume a prefixed lag ℓ between two market return series; instead, the alignment between the two series is dynamically determined by the algorithm. Secondly, this model can handle non-synchronised paired series, thus making the analysis of markets with different lengths of return series feasible. We contend that the variation in matched indices from the two paired series effectively identifies the leader and the follower within these pairs.

Four variations of the dynamic time warping algorithm are introduced in Sections 2.1, 2.2, and 2.3. A qualitative and quantitative assessment of their performance in aligning series is presented in Section 2.4. The methodology for utilising these algorithms to construct the lead-lag matrix is detailed in Section 2.5.

2.1 Classical Dynamic Time Warping

In this section, we describe the core idea of classical dynamic time warping (DTW). The setup and explanation in this manuscript mainly follow the one in [50, page 70]. Given two univariate time series $\mathcal{P}$ and $\mathcal{Q}$ with lengths $\mathcal{L}_\mathcal{P} \in \mathbb{N}$ and $\mathcal{L}_\mathcal{Q} \in \mathbb{N}$ respectively, that is $\mathcal{P} = (p_1, p_2, \cdots, p_{\mathcal{L}_\mathcal{P}})^T$ and $\mathcal{Q} = (q_1, q_2, \cdots, q_{\mathcal{L}_\mathcal{Q}})^T$, the objective of DTW is to find the optimal mapping between them.

To compare the two sequences, one needs a local cost measure.

Definition 2.1.1 (Local cost measure)**.** [14, Definition 1, page 2366] Given a non-empty feature space $\mathcal{F}$, the local cost measure is a function

$$c_{\text{local}} : \mathcal{F} \times \mathcal{F} \to \mathbb{R}_{\geq 0},$$

that satisfies:

1. $c_{\text{local}}(p, q) \geq 0$ for every p, q in $\mathcal{F}$.
2. $c_{\text{local}}(p, q) = 0$ if and only if $p = q$.
3. $c_{\text{local}}(p, q) = c_{\text{local}}(q, p)$ for every p, q in $\mathcal{F}$.
4. $c_{\text{local}}(p, q) \leq c_{\text{local}}(p, r) + c_{\text{local}}(r, q)$ for every p, q and r in $\mathcal{F}$.

It is a positive measure of dissimilarity between two points. Typically, $c(p, q)$ is small if p and q are similar; otherwise, $c(p, q)$ is large. If we calculate the local cost between all pairwise elements of $\mathcal{P}$ and $\mathcal{Q}$, we obtain the local cost matrix

$$C_{\text{local}} \in \mathbb{R}^{\mathcal{L}_\mathcal{P} \times \mathcal{L}_\mathcal{Q}} \text{ with } C_{\text{local}}(i, j) := c(p_i, q_j).$$

Common choices of such measures include Manhattan distance or Euclidean distance [61], that is given two vectors $X \in \mathbb{R}^n$ and $Y \in \mathbb{R}^n$:

$$\text{Manhattan distance: } c(X,Y) = \sum_{i=1}^{n} |X_i - Y_j|$$

$$\text{Euclidean distance: } c(X,Y) = \sqrt{\sum_{i=1}^{n} (X_i - Y_j)^2}$$

These two measures are equivalent when both X and Y lie on the real line $\mathbb{R}$, which is the case in our study, as we aim to calculate the pairwise distances between each pair of points from the one-dimensional series $\mathcal{P}$ and $\mathcal{Q}$. Without loss of generality, we choose the Euclidean distance as our local cost measure.

Based upon the local cost matrix, the DTW algorithm searches for the best alignment between $\mathcal{P}$ and $\mathcal{Q}$ based on a warping path, denoted as $\mathcal{W}\{\mathcal{P}, \mathcal{Q}\}$. Intuitively, a warping path,

$$\mathcal{W}\{\mathcal{P}, \mathcal{Q}\} = \{w_1, \cdots, w_n, \cdots, w_L\} \quad \max(\mathcal{L}_\mathcal{P}, \mathcal{L}_\mathcal{Q}) \leq L \leq \mathcal{L}_\mathcal{P} + \mathcal{L}_\mathcal{Q} - 1, \tag{2.1.1}$$

is a sequence where each element $w_k = (i,j)_k$ indicating that, at the k^{th} step in the warping path, p_i should be aligned with q_j. There are three main constraints to the warping path [50, Definition 4.1, page 70]:

- Boundary conditions: $w_1 = (1,1)$ and $w_L = (\mathcal{L}_\mathcal{P}, \mathcal{L}_\mathcal{Q})$. This ensures that the first and last elements in $\mathcal{P}$ and $\mathcal{Q}$ are matched respectively. This boundary condition can, however, be partly relaxed. We refer interested readers to [61] for details.
- Monotonicity condition: Given $w_l = (i_l, j_l)$ for $l \in [1, \cdots, L]$, $i_1 \leq i_2 \leq \cdots \leq i_L$ and $j_1 \leq j_2 \leq \cdots \leq j_L$. This condition enforces the mapping in chronological order.
- Step size condition: $w_{l+1} - w_l \in \{(1,0), (0,1), (1,1)\}$ for $l \in [1, \ldots, L-1]$. This is sometimes called the continuity constraint. It allows only adjacent cell transitions within the warping path.

Following [45], the step size constraint not only implies the monotonicity constraint but also, in combination with the boundary constraint, make sure that no element in $\mathcal{P}$ and $\mathcal{Q}$ can be omitted and there is no duplication in the mapping alignment.

The cost of a warping path $\mathcal{W}\{\mathcal{P}, \mathcal{Q}\}$ is defined as

$$c_\mathcal{W}(\mathcal{P}, \mathcal{Q}) := \sum_{l=1}^{L} c_{\text{local}}(p_{i_l}, q_{j_l}).$$

Among the various warping paths, the optimal path $\mathcal{W}^*$, which is also the best alignment, follows a path of minimal expense through the cost matrix C_{local}. The DTW distance $\text{DTW}(\mathcal{P}, \mathcal{Q})$, between $\mathcal{P}$ and $\mathcal{Q}$ is then defined as

$$\begin{aligned} \text{DTW}(\mathcal{P}, \mathcal{Q}) &:= c_{\mathcal{W}^*} \\ &= \min\{c_\mathcal{W}(\mathcal{P}, \mathcal{Q}) | \mathcal{W} \text{ is a warping path satisfying necessary constraints}\}. \end{aligned}$$

The optimal warping path can be computed by a dynamic programming algorithm. We start with defining the prefix sequence of $\mathcal{P}$ and $\mathcal{Q}$ as $\mathcal{P}(1:n) := (p_1, p_2, \cdots, p_n)$ for $n \in [1, \cdots, \mathcal{L}_\mathcal{P}]$, and $\mathcal{Q}(1:m) := (q_1, q_2, \cdots, q_m)$ for $m \in [1, \cdots, \mathcal{L}_\mathcal{Q}]$. We then set the accumulated cost matrix with values

$$D(n,m) := \text{DTW}(\mathcal{P}(1:n), \mathcal{Q}(1:m)).$$

Table 2.1 Algorithm for finding the optimal warping path by dynamic programming

Require: Time series $\{X_i\}_{i=1}^{m}$ and $\{Y_j\}_{j=1}^{n}$ with length m and n respectively.
Ensure: Dynamic time warping cost matrix D with shape (m, n).

```
Initialise an empty matrix D with shape (m, n).
for i ← 1 to m do
    D[i][1] = Σ_{k=1}^{i} c_local(X_k, Y_1)
end for
for j ← 1 to n do
    D[1][j] = Σ_{k=1}^{j} c_local(X_1, Y_k)
end for
for i ← 2 to m do
    for j ← 2 to n do
        D[i][j] = min{D[i−1][j−1], D[i−1][j], D[i][j−1]} + c_local(X_i, Y_j)
    end for
end for
return D[m][n]
```

The algorithm of dynamic programming recursion is summarised in Table 2.1. This search has time complexity $\mathcal{O}(\mathcal{L}_\mathcal{P}\mathcal{L}_\mathcal{Q})$ [50, Theorem 4.3, page 72].

To fix ideas, suppose we aim to find the optimal mapping between the time series $\mathcal{P}$ and $\mathcal{Q}$ shown in Figure 2.1(a). We illustrate their local cost matrix in Figure 2.1(b) where, without loss of generality, we consider the Euclidean distance. Figure 2.1(c) shows the cost matrix and the optimal warping path, while Figure 2.1(d) gives the corresponding alignment of the two series.

The warping path plot in Figure 2.1(c) also sheds some light on the leading and lagging relationship between the two sequences. The black dashed line starting at $(1, 1)$ corresponds to the $i = j$ for $i, j \in \min\{\mathcal{L}_\mathcal{P}, \mathcal{L}_\mathcal{Q}\}$. This is an alignment where elements of the time series are mapped to each other at equivalent indices. If the optimal warping path, shown in red, falls below the black dashed line—specifically within the bottom-left region bounded by the grey dashed line—it signifies that the sequence on the vertical axis leads the sequence on the horizontal axis, and vice versa. The same lead-lag relationship is also observable in Figure 2.1(d): for elements to the left of the grey dashed line, sequence $\mathcal{Q}$ demonstrates an increasing or decreasing trend before sequence $\mathcal{P}$ follows; conversely, on the right side, sequence $\mathcal{Q}$ responds to the trend of sequence $\mathcal{P}$.

2.2 Derivative Dynamic Time Warping

As previously mentioned, classical Dynamic Time Warping (DTW) aligns points from two time sequences based on a local cost measure, typically chosen as Manhattan or Euclidean distance. This alignment relies solely on their coordinate values. However, it faces difficulties when two time sequences have similar local shapes but differ in their values. An example is presented in Figure 2.2. Two identical sequences shown in Figure 2.2(a) produce a perfect one-to-one alignment, as illustrated in Figure 2.2(b). However, if we slightly alter their coordinate values, as shown in Figure 2.2(c), by sharpening the peak and deepening the trough, the DTW alignment in Figure 2.2(d) demonstrates a scenario where a single point on one time series is mapped to several points on the other time series. Examples of these are marked in red boxes. This undesirable behaviour is referred to as 'singularities'. According to [36, page 2], this occurs because the algorithm may attempt to account for variability in the Y-axis by warping the X-axis. Although one could, and in fact should, as stated by [57], always perform Z-normalisation to convert

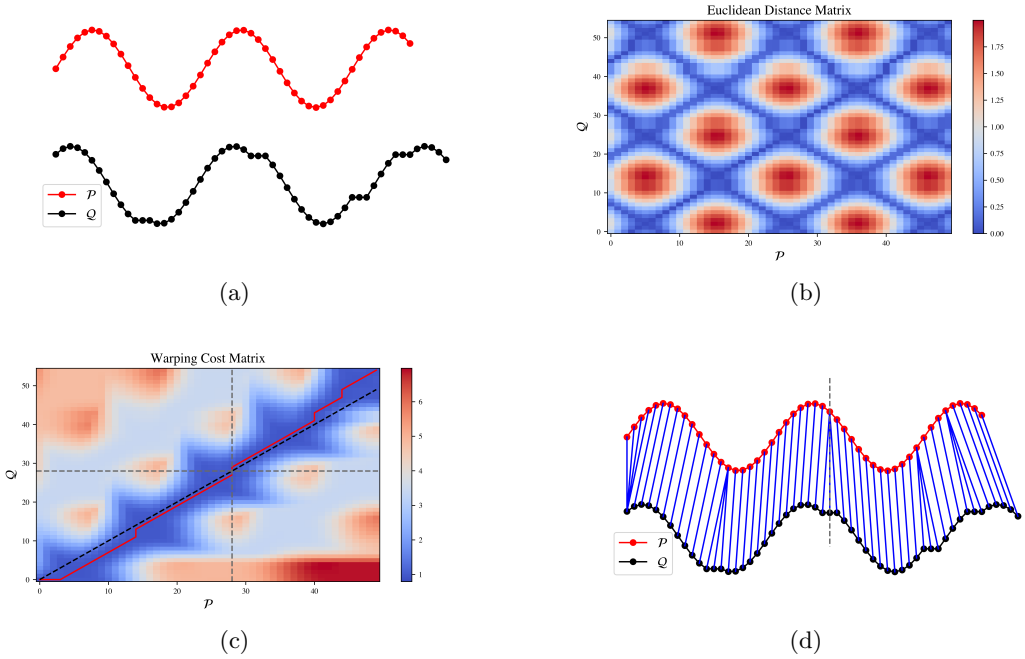


Figure 2.1: Stages of Dynamic Time Warping Alignment.

the time sequences to a common and comparable scale with a mean of 0 and a standard deviation of 1, this approach does not resolve the alignment issue. Consider the scenario where a point p_i from sequence $\mathcal{P}$ has an identical value to q_j from sequence $\mathcal{Q}$, yet the neighbourhood of p_i is in a rising trend while the neighbourhood of q_j is in a falling trend. DTW may map these points onto one another to achieve minimal overall cost. Intuitively, however, such mappings should be avoided, especially when the lead-lag relationship is important.

To address this problem, [36] modifies DTW, denoted as DDTW. Instead of finding the optimal warping based on the raw values of the sequences, we consider the estimated local derivative of the sequence. The derivative of points on sequence $\mathcal{P}$ is estimated by the following equations

$$\begin{aligned} &D_{\mathcal{P}}[p_n] = \frac{(p_n - p_{n-1}) + ((p_{n+1} - p_{n-1})/2)}{2}, \text{ with } n \in [2, \cdots, \mathcal{L}_{\mathcal{P}} - 1], \\ &D_{\mathcal{P}}[p_1] = D_{\mathcal{P}}[p_2], \text{ and} \\ &D_{\mathcal{P}}[p_{\mathcal{L}_{\mathcal{P}}}] = D_{\mathcal{P}}[p_{\mathcal{L}_{\mathcal{P}}-1}]. \end{aligned} \tag{2.2.1}$$

This is effectively the mean value between the slope of the line from the left neighbour to the point and the slope of the line from the left neighbour to the right neighbour. [36] suggests that after replacing the original sequence with the estimated derivative, the following procedure is the same as the classical dynamic time warping algorithm. DDTW also outperforms DTW in the simulated alignment tests in [36].

The procedure to compute the DDTW algorithm has a time complexity of $\mathcal{O}(\mathcal{L}_{\mathcal{P}}\mathcal{L}_{\mathcal{Q}})$, with some added constant factors because of the addition derivative estimation step.

2.3 Shape Dynamic Time Warping

While DDTW considers the slope of the time series, it only considers the slope within a local neighbourhood, failing to consider the global features. An improvement was proposed

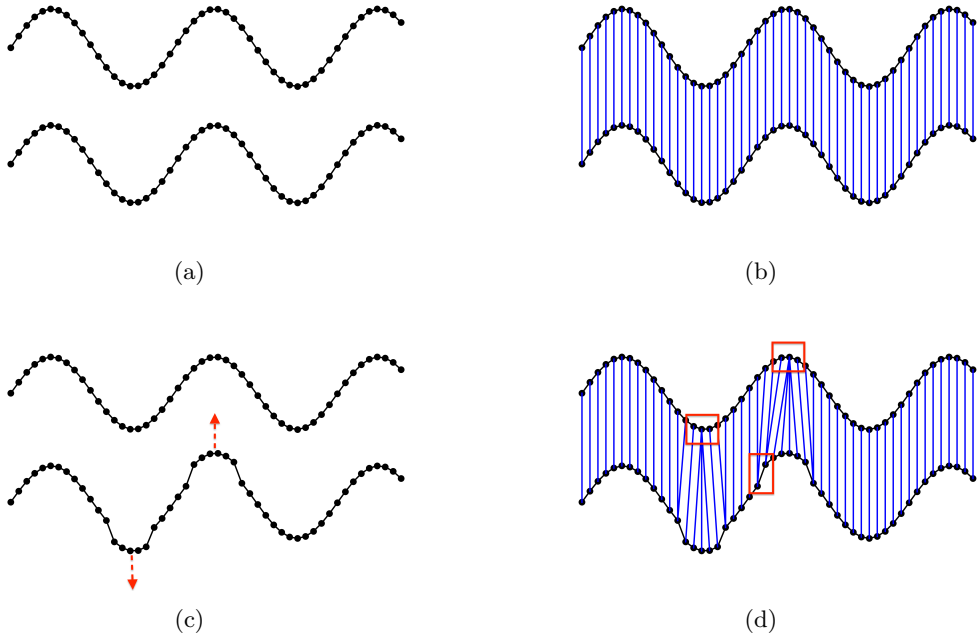


Figure 2.2: Illustration of the Singularity Problem in classical Dynamic Time Warping.

in [73] by dealing with multidimensional time series to account for both global features and local shapes, known as shape dynamic time warping (shapeDTW). The intuitive idea is to convert a one-dimensional time series $\mathcal{P} = (p_1, p_2, \cdots, p_{\mathcal{L}_\mathcal{P}})$ of length $\mathcal{L}_\mathcal{P}$ into a multidimensional series $\mathcal{D} = (d_1, \cdots, d_{\mathcal{L}_\mathcal{P}}) \in \mathbb{R}^{\mathcal{L}_\mathcal{P} \times l}$, where each subsequence d_i of length l embeds the information of the point p_i. The dependent multidimensional DTW algorithm proposed in [60] is then applied to the two multidimensional series to calculate the distance cost and determine the optimal warping path.

We now introduce the execution of shapeDTW in detail. Given a univariate time series of length $\mathcal{L}_\mathcal{P}$, e.g., $\mathcal{P} = (p_1, p_2, \ldots, p_{\mathcal{L}_\mathcal{P}})$, the first step is to extract the subsequence d_i of length l from each temporal point p_i. The subsequence d_i is called the descriptor and is centred on p_i. As suggested by [73], the window l is usually much smaller than $\mathcal{L}_\mathcal{P}$, and typically, $l \bmod 2 = 1$ so that p_i is centred. The resulting descriptor looks like this:

$$d_i = \left(p_{i-\frac{l-1}{2}}, \ldots, p_i, \ldots, p_{i+\frac{l-1}{2}}\right).$$

To obtain subsequences of equal length for each point, both ends of $\mathcal{P}$ need to be padded by $\lfloor \frac{l}{2} \rfloor$ with duplicates of p_1 and $p_{\mathcal{L}_\mathcal{P}}$, respectively.

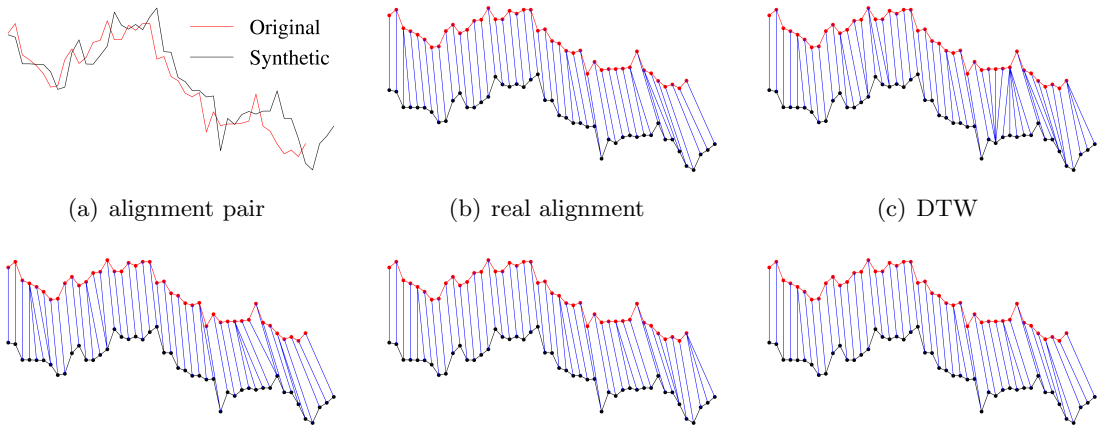
The shape descriptor d_i is expected to encode local structural information around the temporal point p_i [73]. Here, we examine two types of descriptors mentioned in [73]. The first approach is to let the subsequence act as the local shape descriptor. Unlike DDTW, each subsequence contains information from the series over a window of l points, thereby capturing structural information over a longer time window. Another approach is to apply the DDTW (2.2.1) to each subsequence, thereby achieving a y-shift. We refer to this method as shapeDTW derivative (shapeDDTW). Once both multidimensional series are converted, [64] suggests calculating the distance matrix by applying a chosen distance measure to each pair of subsequences, followed by using dynamic programming in Table 2.1, to search for the optimal warping path. Since the descriptors contain near-synchronised information, this method aggregates the information distributed across different dimensions and examines the differences over time.

Here are some further remarks on shapeDTW. If one sets $l = 1$, the problem reduces to DTW or DDTW depending on the chosen descriptors. [61] mentions that by combining the steps of extracting the descriptors and running the DTW alignment, shapeDTW and shapeDDTW have a time complexity of $\mathcal{O}(\mathcal{L}_\mathcal{P} \times \mathcal{L}_\mathcal{Q} \times l)$ for time series $\mathcal{P}$ and $\mathcal{Q}$.

2.4 Comparative Analysis of DTW Techniques

In this section, we analyse the alignments of the four algorithms introduced above qualitatively and quantitatively. To compare the alignment results with known ground truth, we simulate aligned pairs based on the simulation method proposed in [73]. The intuition behind this simulation is as follows: given a time series $\mathcal{T}$ of length L, we first perturb it using a random noise vector of equal length $\mathcal{S}$. Then, we randomly select α per cent of the points from $\mathcal{T}$ and duplicate each of them by a fixed amount τ. The resulting sequence $\mathcal{T}'$ has a length of $L + \lfloor \alpha L \rfloor \tau$, with each value slightly perturbed by noise. Under these settings, we can control the level of similarity between the original and perturbed time series and also have the ground truth warping path to compare against.

Figure 2.3 includes the warping path found by different DTW algorithms. The correct warping path for the pair in Figure 2.3(a) is shown in Figure 2.3(b). Compared to the DTW alignment in Figure 2.3(c), DDTW alignment in Figure 2.3(d) eases the singularity problems as discussed above, whereas it still mismatches some pairs compared to the ground truth. The alignments found by shapeDTW and shapeDDTW in Figure 2.3(e) and 2.3(f) achieve much closer alignment with the ground truth.



(a) alignment pair (b) real alignment (c) DTW

(d) DDTW (e) shapeDTW (f) shapeDDTW

Figure 2.3: Different dynamic time warping alignments between actual and simulated pairs for the HKG Hang Seng Index Futures: September to October 2018.

To quantitatively assess the alignment, we consider two fundamental metrics. The first is proposed in [36] to evaluate the extent of warping. Given two time series with lengths $\mathcal{L}_\mathcal{P}$ and $\mathcal{L}_\mathcal{Q}$, the number of warping paths K is bounded by $\max(\mathcal{L}_\mathcal{P}, \mathcal{L}_\mathcal{Q}) \leq K < \mathcal{L}_\mathcal{P} + \mathcal{L}_\mathcal{Q} - 1$. We define the Singularity Severity Score W as

$$W = \frac{K - \max(\mathcal{L}_\mathcal{P}, \mathcal{L}_\mathcal{Q})}{\max(\mathcal{L}_\mathcal{P}, \mathcal{L}_\mathcal{Q})}.$$

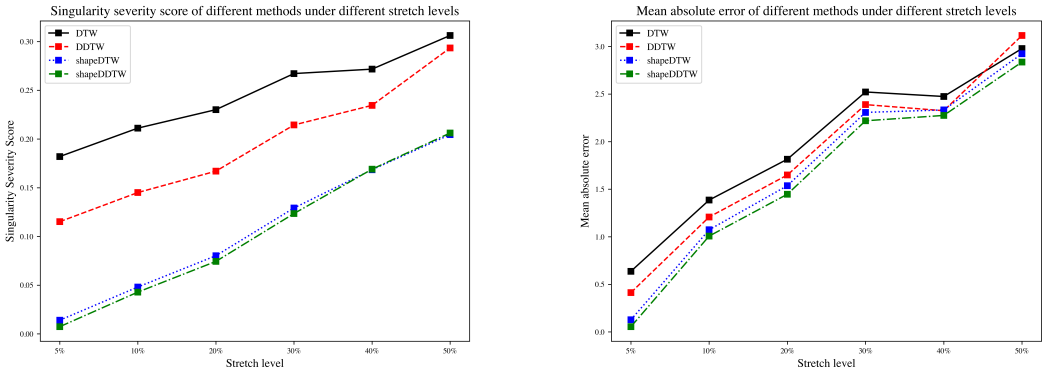
W ranges from 0 to 1, with higher values indicating a greater number of singularities in the warping process. We also define another metric, the Mean Absolute Deviation (MAE), as proposed in [37]. This metric assesses the alignment between two warping paths—the ground truth and the DTW model results—represented as a series of index pairs (i, j).

This indicates that the i-th index from the first series should map to the j-th index from the second series. In cases where multiple points are mapped to a single index j, only the last mapping of any index to index j is retained. Mathematically, it is expressed as

$$\text{MAE} = \frac{1}{N}\sum_{k=1}^{N}\delta_k,$$

where $\delta_k = |j_{\text{true},k} - j_{\text{dtw},k}|$, and N is the number of retained warping pairs. This is a measure of the precision of the models' alignment.

By applying the simulation algorithm to 180 different series to create 180 aligned pairs, we test the singularity severity score and MAE for four models. We gradually increase the stretch level α and adjust the noise magnitude $\|\mathcal{S}\|_2$ correspondingly. A higher stretch level indicates a greater number of misordered pairs. The scores are presented in Figure 2.4(a). It can be observed that the two multidimensional DTW methods, shapeDTW and its variation, shapeDDTW, effectively reduce the problem of singularity better than the DDTW and DTW at all disorder levels. Based on the MAE plot in Figure 2.4(b), shapeDTW and shapeDDTW also consistently outperform DDTW and DTW over various disorder levels. Especially when there is minimal stretch and noise, shapeDTW and shapeDDTW achieve a halving of the MAE compared to DTW, approaching values close to zero, which indicates almost perfect matching. This suggests that the multidimensional dynamic time warping algorithm is more effective at matching two series based on their degree of similarity.



(a) (b)

Figure 2.4: Quantitative Comparison of Singularity Severity and Mean Absolute Error Across Dynamic Time Warping Models for Simulated Paired Series with Various Levels of Disorder.

2.5 Computation of Lead-Lag Matrix Using Dynamic Time Warping Models

In this section, we outline our method for calculating the lead-lag metric matrix. Consider the return series of market m from time $t = 1$ to $t = T$, $X_m = (X_{1,m}, \ldots, X_{t,m}, \ldots, X_{T,m})$, if it contains a zero at some time t, it implies that market m was not traded at that time; such points are removed from the series, leveraging the dynamic time warping algorithm's capability to handle non-synchronised series. Each market return series is normalised to zero mean and unit variance, as discussed in [61], following the conventions in DTW research.

To identify the leader and follower and the relative lag between series X_m and X_n, we first compute the difference in each aligned index pair (i, j) in their warping path $\mathcal{W}\{X_m, X_n\}$ defined in (2.1.1), denoted as:

$$\Delta\mathcal{W}\{X_m, X_n\} = \{\Delta w_1, \ldots, \Delta w_k\},$$

where $\Delta w_k = \Delta(i, j)_k = j - i$. The relative lags between X_m and X_n, $\gamma\{X_m, X_n\}$, are typically estimated by mode or median according to [72], which can be expressed as:

$$\gamma\{X_m, X_n\} = \begin{cases} \text{Mode}(\Delta\mathcal{W}\{X_m, X_n\}) & \text{if using Mode} \\ \text{Median}(\Delta\mathcal{W}\{X_m, X_n\}) & \text{if using Median} \end{cases}$$

If being selected by mode and $\gamma\{X_m, X_n\} > 0$, it indicates that most of the time $\Delta w_k > 0$, implying that a later index j from X_n is aligned to an earlier index i in X_m, thus X_m leads X_n.

We now present the algorithms for one-dimensional dynamic time warping (DTW and DDTW) as described in Table 2.2 and for multi-dimensional dynamic time warping (shapeDTW and shapeDDTW) in Table 2.3. The optimal warping path is computed using the Python library DTAIDistance [47].

Table 2.2 Algorithm for using DTW and DDTW to identify lead-lag relationship

Require: Feature matrix $\mathbf{X}_t \in \mathbb{R}^{T \times M}$ representing market returns observed at time t, where T is the length of lookback days, and M is the number of markets.
Ensure: Lead-lag metric matrix $\mathbf{V}_t \in \mathbb{R}^{M \times M}$

Initialise an empty matrix $\mathbf{V}_t$ with shape (M, M).
for $i \leftarrow 1$ to M **do**
  for $j \leftarrow i + 1$ to M **do**
    Extract paired return series of market i and j, X_i and X_j, both with length T.
    Remove the zero values from the series, resulting in X_i' and X_j' respectively.
    if DDTW **then**
      Calculate the local slope of each series using equation (2.2.1), and replace the values in X_i' and X_j' with their derivatives.
    end if
    Normalise X_i' and X_j' to zero mean and unit variance.
    Compute the optimal warping path $\Delta\mathcal{W}\{X_i', X_j'\}$ using the dynamic programming algorithm in Table 2.1.
    Calculate the differences $\Delta\mathcal{W}\{X_i', X_j'\}$.
    Determine the relative lag $\gamma\{X_i', X_j'\}$ by taking the mode or median of $\Delta\mathcal{W}\{X_i', X_j'\}$.
    Set $\mathbf{V}_t[i][j] = \gamma\{X_i', X_j'\}$
    Set $\mathbf{V}_t[j][i] = -\gamma\{X_i', X_j'\}$
  end for
end for
return $\mathbf{V}_t$

Table 2.3 Algorithm for using shapeDTW and shapeDDTW to identify lead-lag relationship

Require: Feature matrix $\mathbf{X}_t \in \mathbb{R}^{T \times M}$ representing market returns observed at time t, where T is the length of lookback days, and M is the number of markets.

Require: Length of the descriptor in ShapeDTW algorithm l, where l should be an odd positive integer.

Ensure: Lead-lag metric matrix $\mathbf{V}_t \in \mathbb{R}^{M \times M}$

Initialise an empty matrix $\mathbf{V}_t$ with shape (M, M).
for $i \leftarrow 1$ to M **do**
  for $j \leftarrow i + 1$ to M **do**
    Extract paired return series of market i and j, X_i and X_j, both with length T.
    Remove the zero values from the series, resulting in X_i' and X_j' respectively.
    Convert X_i' and X_j' to multi-dimensional descriptor series $\mathbf{X}_i$ and $\mathbf{X}_j$.
    Pad the beginning and end of each series by repeating the first and last element $\frac{l-1}{2}$ times.
    At each original index i, extract the subsequence of length l centred at i.
    Take the subsequence as a vector for describing the respective index.
    if shapeDDTW **then**
      Apply the formula of DDTW (2.2.1) to each dimension in each series respectively to calculate the local shape of the descriptors and replace the original value with the descriptors.
    end if
    Normalise each dimension in each series to have zero mean and unit variance.
    Compute the optimal warping path $\Delta\mathcal{W}\{\mathbf{X}_i, \mathbf{X}_j\}$ using a dynamic programming algorithm in Table 2.1 by treating each dimension as one data point.
    Calculate the differences $\Delta\mathcal{W}\{\mathbf{X}_i, \mathbf{X}_j\}$.
    Determine the relative lag $\gamma\{\mathbf{X}_i, \mathbf{X}_j\}$ by taking the mode or median of $\Delta\mathcal{W}\{\mathbf{X}_i, \mathbf{X}_j\}$.
    Set $\mathbf{V}_t[i][j] = \gamma\{\mathbf{X}_i, \mathbf{X}_j\}$
    Set $\mathbf{V}_t[j][i] = -\gamma\{\mathbf{X}_i, \mathbf{X}_j\}$
  end for
end for
return V_t

Chapter 3

Graph Learning

In this chapter, we introduce the method used to convert the pairwise lead-lag relationship into the 'network momentum'. This network is expected to reflect the interconnected relationship in the cross-sectional momentum between paired markets. Effectively, we transform the lead-lag matrix, identified by the algorithms proposed in Sections 1.3 and 2.5, into an adjacency matrix with non-negative values which can be interpreted as weights for propagating time series momentum from one market to its connected markets. We refer to this matrix as the network momentum matrix.

We begin by defining prerequisite definitions and knowledge on graph learning in Section 3.1, the model used for approximating the adjacency matrix in Section 3.2, and the algorithm for applying it to the lead-lag matrix is summarised in Section 3.3.

3.1 Graph Learning Preliminaries

A graph, denoted as $G(V, E)$, is defined as a combinatorial object on two sets: a set of nodes $V = \{i\}_{i=1}^{N}$ and a set of edges $E = \{(i, j)\}_{i \sim j}$. A node is usually a vector in $p-$dimensional space for a data point with the form

$$x^{(i)} = \begin{pmatrix} x_1^{(i)} \\ \vdots \\ x_p^{(i)} \end{pmatrix},$$

and a graph with N nodes can therefore be summarised in the matrix form

$$X_{N \times p} = \begin{pmatrix} {x^{(1)}}^T \\ \vdots \\ {x^{(N)}}^T \end{pmatrix}.$$

It is important to study the underlying topology of the graph network, that is we want to introduce some distance metrics to calculate the similarity or dissimilarity between nodes. In fact, this is an important determinant of the dataset in the unsupervised learning methods, the pairwise interaction is the insight that leads into the graph representation of the data. Based on the distance matrix, we can determine which two nodes should be linked and the edge between nodes i and j belongs to the set E.

The graph can be equivalently represented in an unweighted adjacency matrix $A_{N \times N}$, where

$$A_{ij} = \begin{cases} 1, & \text{if nodes } i \text{ and } j \text{ are connected} \\ 0, & \text{otherwise.} \end{cases}$$

In the cases where each edge has an associated weight, we could construct the weighted adjacency matrix by replacing 1 with $w_{i,j} \in \mathbb{R}$ if nodes i and j are connected. In either case, such an adjacency matrix is symmetric, that is $A = A^T$.

The degree vector d of a graph is defined by

$$d = A\mathbf{1},$$

where A is the adjacency matrix of the graph, and $\mathbf{1}$ is a column vector of ones, with each entry d_i of d representing the number of edges the corresponding node i has with other nodes. The degree matrix D is then defined as

$$D = \text{diag}(d),$$

where $\text{diag}(d)$ creates a diagonal matrix with the elements of $\mathbf{d}$ on its diagonal, and all off-diagonal elements are zero.

The Combinatorial Graph Laplacian, in short the Laplacian matrix L, is defined as

$$L = D - A.$$

Next, we define the Dirichlet energy which is commonly used as a measure of smoothness of a graph.

Definition 3.1.1 (Dirichlet energy)**.** [33, page 5] Given a graph G on a matrix $X \in \mathbb{R}^{N\times p}$ with node vectors denoted as $x^{(i)}$, $x^{(j)}$, and an adjacency matrix $A \in \mathbb{R}^{N\times N}$, the Dirichlet energy of X on G is defined as:

$$\|X\|^2_{\mathcal{D},G} = \frac{1}{2}\sum_{i,j} \|x^{(i)} - x^{(j)}\|_2^2 A_{i,j}.$$

The matrix X is called smooth on graph G if its Dirichlet energy is small. This smoothness is typically achieved by only connecting nodes i and j when $x^{(i)}$ and $x^{(j)}$ are similar, leading to sparsity in A. Fewer edges are connected, which intuitively reduces the matrix's Dirichlet energy. The underlying concept and further implications of this sparsity are discussed in Section 3.2. The smoothness ultimately measures the extent to which the matrix X varies in response to the assumptions about variability represented by the weights A. This can also be considered a cost function, and in order to achieve a smooth graph, two nodes should only be connected if they have similar values. This can be further written as

$$\|X\|^2_{\mathcal{D},G} = \text{tr}(X^T L X), \tag{3.1.1}$$

where we present the derivation in Appendix A.2.

3.2 From Data to Structure

A well-studied topic in graph learning involves learning a graph $G(V, E)$ with $|V| = N$ nodes from a noisy dataset containing N observations, each represented by a p-dimensional vector $x^{(i)}$. The objective is to ensure that the observations in X exhibit smoothness on the graph G with adjacency matrix A, as indicated by a small Dirichlet energy in Definition 3.1.1. One rationale behind this objective is that smooth signals admit low pass band-limited properties [46, page 25], and the graph learning task can thus be conceptualised as finding efficient information processing transforms for graph signals. For a formal formulation and detailed interpretation of graph smoothness, we direct interested readers to [46].

One straightforward approach to detecting pairwise similarity among nodes involves calculating a coherence measure, such as correlation, Jaccard coefficient, or Euclidean

distance, and then manually setting a threshold to filter edges based on this measure. Alternatively, geometric information can be used, such as connecting nodes within a certain distance ϵ, or mathematically, within the ϵ-ball. Techniques like k-nearest neighbourhood or the minimum spanning tree have also been explored. A significant limitation of these methods, as highlighted in [46, page 22], is the absence of a validation framework, making it challenging to verify whether the resulting graph accurately reflects the underlying structure. Moreover, the adjacency matrix A may vary significantly with minor adjustments in parameters like ϵ.

To address these challenges, one strategy is to explore the entire space of potential candidate Laplacian matrices $\mathcal{L}$, defined as

$$\mathcal{L} = \{L \in \mathbb{R}^{N\times N} : L_{i,j} = L_{j,i} \leq 0 \text{ for } i \neq j, \text{ and } L_{i,i} = -\sum_{j\neq i} L_{i,j}\}.$$

As suggested in [33], it is more intuitive and also equivalent to search for a valid weighted adjacency matrix A from the space

$$\mathcal{A} = \{A \in \mathbb{R}_+^{N\times N} : A = A^T, \operatorname{diag}(A) = 0\}.$$

Pursuing our goal of minimising Dirichlet energy, a practical approach is to tackle the optimisation problem:

$$\text{minimise}_{L\in\mathcal{L}} \operatorname{tr}(X^T L X). \tag{3.2.1}$$

However, as mentioned in [34, page 922], problem (3.2.1) would naturally lead to a very sparse graph, where each node is connected to only a few other nodes compared to the total number of nodes in the graph. To see this, let us define a pairwise distance matrix $Z \in \mathbb{R}_+^{N\times N}$, and

$$Z_{i,j} = \|x^{(i)} - x^{(j)}\|_2^2.$$

Therefore the Dirichlet energy in Definition 3.1.1 can be equivalently expressed as

$$\|X\|_{\mathcal{D},G}^2 = \frac{1}{2}\sum_{i,j} Z_{i,j}A_{i,j} = \frac{1}{2}\|A \odot Z\|_1, \tag{3.2.2}$$

where $\odot$ denotes the element-wise product. Formula (3.2.2) shows that the Dirichlet energy is a weighted ℓ_1 norm of the adjacency matrix A, and minimising it in problem (3.2.1) means penalising edges connecting distant rows of X. The ideal graph learning model should allow us to control the sparsity of the graph. It is important to note that the problem (3.2.1) can also result in isolated nodes—nodes not connected to any others in the graph. If one wants a fully connected graph where every node links to at least one neighbour, such isolation should be avoided.

To fulfil the two desiderata mentioned above, an alternative approach is proposed in [34, page 923] to learn the adjacency matrix and, consequently, the Laplacian matrix by solving an optimisation problem.

Definition 3.2.1 (Graph learning model)**.** [34, Page 923] Given a smooth matrix $X \in \mathbb{R}^{N\times p}$ on a graph G, and sparsity parameters $\alpha > 0$ and $\beta \geq 0$, can be found by solving the following convex optimisation problem:

$$\begin{aligned} &\text{minimise}_{A\in\mathcal{A}} \quad \operatorname{tr}(X^T(D-A)X) - \alpha \mathbf{1}^T \log(A\mathbf{1}) + \beta\|A\|_F^2 \\ &\text{such that} \quad A_{i,j} = A_{j,i}, \quad A_{i,j} \geq 0 \quad \forall i \neq j, \quad \operatorname{diag}(A) = \mathbf{0}. \end{aligned}$$

Here, a logarithmic penalty term is used to prevent isolated nodes so that every node has at least one connected neighbour, and the Frobenius norm is used to control sparsity,

unlike the ℓ_1 norm, it only penalises the edge with large magnitude without disproportionately affecting smaller edges.

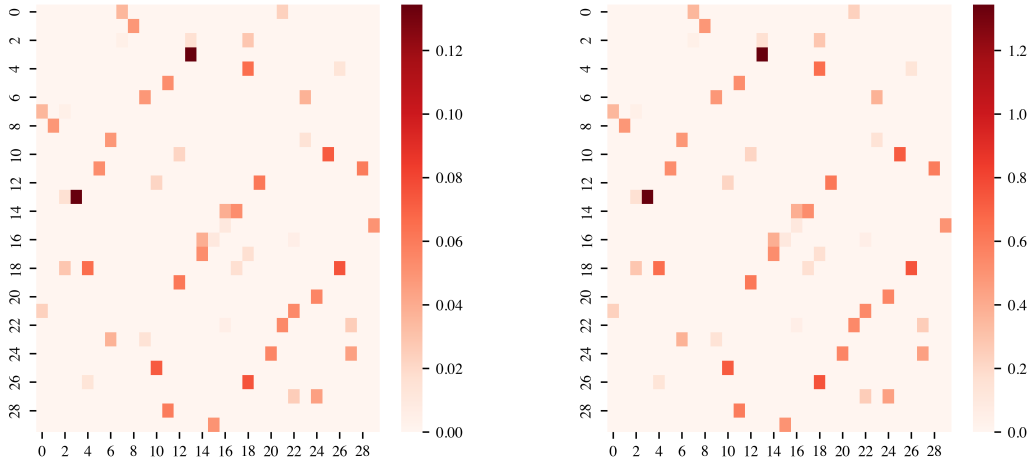
An interesting result in [34] says that changing α only changes the scale of the solution, not the sparsity pattern. To formulate this finding, let us define the following proposition.

Proposition 3.2.2. *[34, Proposition 2, page 923] Let $F(X, \alpha, \beta)$ denote the solution of our model 3.2.1 for input signal X and parameters α and β. Then the following property holds for any $\gamma > 0$:*

$$F(X, \alpha, \beta) = \alpha F(X, 1, \alpha\beta).$$

Proof. See [34, Proposition 2, page 923]. □

This proposition says that for two distinct values, $\alpha_1 > 0$ and $\alpha_2 > 0$, if we select β_1 and β_2 such that the product $\alpha_1\beta_1 = \alpha_2\beta_2$, then the resulting adjacency matrices A_1 and A_2 will be related by $A_1 = \frac{\alpha_1}{\alpha_2}A_2$. We illustrate this relationship in Figure 3.1 using two sets of hyperparameters: $(\alpha, \beta) = (1, 1)$ and $(10, 0.1)$ respectively. The sidebars indicate that the range changes from 0.12 in Figure 3.1(a) to 1.2 in Figure 3.1(b); however, the color grids in both figures remain consistent. This demonstrates that modifying α and β while maintaining their product constant affects only the magnitude of the edges, not the sparsity of the graph.



(a) (b)

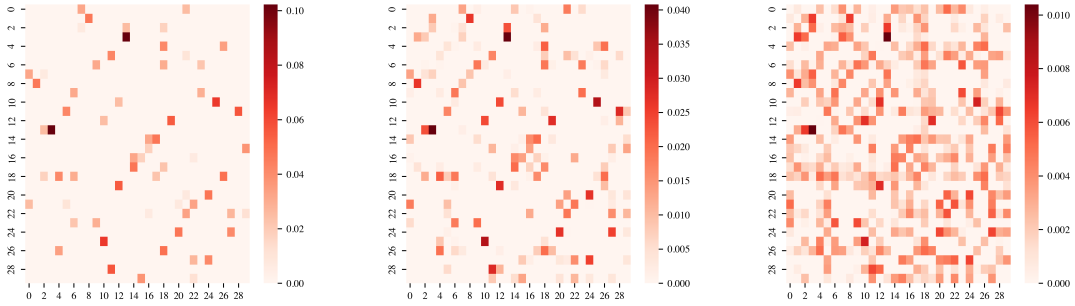
Figure 3.1: Illustration of changing α while keeping $\alpha\beta$ constant only affects the scale of the graph learning model: (α, β) = (1,1) for (a) and (10,0.1) for (b).

In general, as the parameters α and β increase, a denser graph is obtained. This phenomenon is shown in Figure 3.2, where α and β gradually increase from 0.01 to 10, the colour grids become denser from Figure 3.2(a) to Figure 3.2(c).

In addition, the number of edges in the graph, for varying combinations of α and β, can be calculated by

$$E = \frac{1}{2}\sum_{i=1}^{N} d_i,$$

where d_i represents the degree of the $i-$th node. The results, as shown in Figure 3.3, confirm that as α and β increase from 0.1 to 100, the number of edges exceeds 400.



(a) (b) (c)

Figure 3.2: Illustration of larger values of α and β leading to a denser graph.

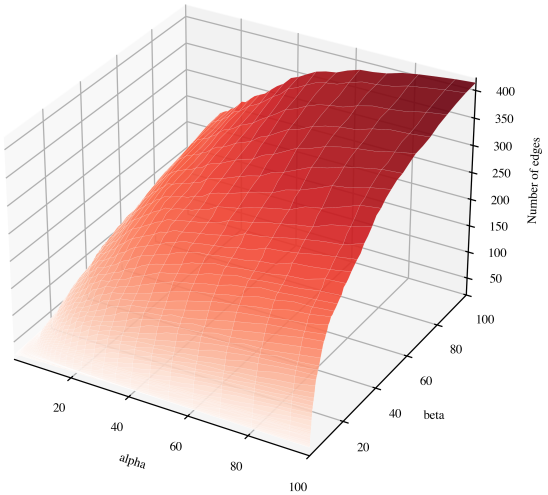


Figure 3.3: Variations in the number of edges for different α and β settings.

3.3 Computation of Network Momentum Matrix Using Graph Learning Model

Given the lead-lag matrix $\mathbf{V}_t$ at trading time t, we replace the signal matrix X in the graph learning model defined in 3.2.1 with $\mathbf{V}_t$ to obtain an adjacency matrix with non-negative edge weights and no isolated markets. The edge values reflect the interconnected relationship of the leadingness of the markets.

It is suggested in [56] that to mitigate the effects of scale differences in constructing network momentum—arising from the variance in the number of connections some nodes have, with some connected to many other assets and others to only a few—a graph normalisation should be applied to the adjacency matrix $\mathbf{A}_t$ before using it to aggregate time series momentum. This normalisation is defined as follows:

$$\tilde{\mathbf{A}}_t = \mathbf{D}_t^{-1/2}\mathbf{A}_t\mathbf{D}_t^{-1/2} \tag{3.3.1}$$

The empirical analysis in [56] suggests that combining S adjacency matrices obtained from different lead-lag matrices $\mathbf{V}_t$ based on historical price data with different lookback windows can improve performance. Therefore, we define the ensemble adjacency matrix $\bar{\mathbf{A}}_t$ as:

$$\bar{\mathbf{A}}_t = \frac{1}{S}\sum_{s=1}^{S}\mathbf{A}_t^{(s)} \tag{3.3.2}$$

and will compare the strategy performance between using and not using the ensemble mechanism.

We now summarise the algorithm for calculating the network momentum matrix in Table 3.1. In practice, the optimisation problem in the graph learning model 3.2.1 is solved numerically with MOSEK and Python library CVXPY [18].

Table 3.1 Algorithm for Computing the Network Momentum Matrix Using Graph Learning

Require: Series of lead-lag matrices $\{\mathbf{V}_t^s \in \mathbb{R}^{M\times M}\}_{s=1}^{S}$ observed at trading time t, where M is the number of markets and S is the number of historical price data inputs, $S \geq 1$.
Require: Hyperparameters $\alpha > 0$ and $\beta \geq 0$ for sparsity control.
Ensure: Normalised network momentum matrix $\tilde{\mathbf{A}}_t \in \mathbb{R}^{M\times M}$.
1: Initialise an ensemble adjacency matrix $\bar{\mathbf{A}}_t$ with zeros of shape (M, M).
2: **for** $s = 1$ to S **do**
3: Replace the signal matrix X in the graph learning model defined in 3.2.1 with $\mathbf{V}_t^s$ to obtain the adjacency matrix $\mathbf{A}_t^s$.
4: Update the ensemble adjacency matrix according to (3.3.2): $\bar{\mathbf{A}}_t \leftarrow \bar{\mathbf{A}}_t + \frac{1}{S}\mathbf{A}_t^s$.
5: **end for**
6: Normalise $\bar{\mathbf{A}}_t$ using the graph normalisation formula (3.3.1) to obtain $\tilde{\mathbf{A}}_t$.
7: **return** $\tilde{\mathbf{A}}_t$
Note: If $S = 1$, this algorithm is equivalent to not using the graph ensemble method, thereby directly applying normalisation to the single obtained adjacency matrix.

Chapter 4

Experiments

4.1 Data

Our raw dataset contains the daily mid-price of the bid and ask prices at the close of 28 futures contracts. The contracts primarily come from the commodity sectors—agriculture, energy, and metals—and include some equity indices. The model training spans from June 2002 to June 2024, with strategy performance evaluated on out-of-sample data from January 2005 to June 2024. However, not every contract feature has data on each day. A complete list of the markets included in our portfolio can be found in Appendix B.1.

To evaluate the robustness and generalisability of our strategy across diverse out-of-sample price data and portfolios, we generate 100 sets of bootstrapped price data from the original dataset. The set of synthetic data is used for additional backtesting. We refer interested readers to Appendix C for motivations behind this resampling method, as well as a detailed description and the setup of the bootstrapping algorithm.

4.2 Set Up for Time Series Momentum Features

In this section, we present the construction of the time series momentum features. We begin by introducing two preliminary definitions.

Definition 4.2.1 (Exponentially Weighted Moving Average)**.** [24, Section 6.4.3.1] Let $\{x_t\}_{t=1}^T$ be a sequence of real-valued observations, and let $N \in \mathbb{Z}_+$ such that $N < T$. The exponentially weighted moving average (EWMA) with a span of N days, $\{\mu_t\}_{t=1}^T$, is defined by the recursive relation:

$$\mu_t := \alpha x_t + (1-\alpha)\mu_{t-1}$$

where μ_0 is the initial condition, typically set to x_1. The smoothing factor α is defined as $\alpha := \frac{2}{N+1}$, and for clarity in different contexts, the EWMA at time t for a specific α is denoted by μ_t^α.

Definition 4.2.2 (Exponentially Weighted Moving Standard Deviation)**.** [56] Let $\{x_t\}_{t=1}^T$ be a sequence of real-valued observations, and let $N \in \mathbb{Z}_+$ such that $N < T$. The exponentially weighted moving standard deviation (EWMstd) with a span of N days at time t, denoted by σ_t, is defined as:

$$\sigma_t := \sqrt{\frac{\sum_{\tau=0}^t w_\tau (x_\tau - \mu_\tau)^2}{\sum_{\tau=0}^t w_\tau}},$$

where the smoothing factor α is defined as $\alpha := \frac{2}{N+1}$, the exponentially decaying weight at time τ, w_τ, is defined as $w_\tau := (1-\alpha)^\tau$, and μ_τ is the EWMA in Definition 4.2.1.

Now we introduce a classical individual momentum feature based on price information. For each market m and time index $t = 1, \ldots, T$, we denote the price for market m at time t as $P_{t,m} \in \mathbb{R}$, and therefore the price time series for market m is denoted as

$$P_m := (P_{1,m}, P_{2,m}, \ldots, P_{T,m}) \in \mathbb{R}^T.$$

We construct $\mathbf{P} \in \mathbb{R}^{T \times M}$, representing a matrix of M market prices across a time horizon of T, where each vector is a price time series for a market.

Definition 4.2.3 (Price delta)**.** Given a market m, denote its price time series from time $t = 1$ to $t = T$ as $(P_{1,m}, P_{2,m}, \ldots, P_{T,m}) \in \mathbb{R}^T$, the price delta for it at time t, $\Delta_{t,m}$, is defined as the first difference in its price series,

$$\Delta_{t,m} := P_{t,m} - P_{t-1,m}.$$

Then, the price delta time series for market m is denoted as

$$\Delta_m := (\Delta_{1,m}, \ldots, \Delta_{T,m}),$$

and the matrix of market price deltas is defined as $\mathbf{\Delta} \in \mathbb{R}^{T \times M}$.

Considering that each market exhibits different levels of price volatility, we choose to normalise the price deltas of each market to have unit volatility. This step aligns with the extant literature [56, 7] in their construction of time series momentum features.

Definition 4.2.4 (Volatility-scaled price delta)**.** Given a market m, denote its price delta time series from time $t = 1$ to $t = T$ as $(\Delta_{1,m}, \ldots, \Delta_{T,m})$, let the exponential weighted moving standard deviation at time t over a span of 22 days denoted as $\sigma^{22}_{t,m}$ following Definition 4.2.2. The volatility scaled price deltas for market m at time t is defined as

$$\tilde{\Delta}_{t,m} := \frac{\Delta_{t,m}}{\sigma^{22}_{t,m}}.$$

The time series of volatility-scaled price delta for market m is denoted as

$$\tilde{\Delta}_m = (\tilde{\Delta}_{1,m}, \ldots, \tilde{\Delta}_{T,m}),$$

and the matrix of all market price deltas is defined as $\tilde{\mathbf{\Delta}} \in \mathbb{R}^{T \times M}$.

Definition 4.2.5 (Volatility-scaled price)**.** Given a market m, denote its volatility-scaled price delta time series from time $t = 1$ to $t = T$ as $(\tilde{\Delta}_{1,m}, \ldots, \tilde{\Delta}_{T,m})$, the volatility-scaled price for market m at time t is defined as

$$\tilde{P}_{t,m} := \sum_{i=0}^{t} \tilde{\Delta}_{i,m}.$$

The time series of volatility-scaled price for market m is denoted as

$$\tilde{P} := (\tilde{P}_{1,m}, \ldots, \tilde{P}_{T,m}),$$

and the matrix of all market price deltas is defined as $\tilde{\mathbf{P}} \in \mathbb{R}^{T \times M}$.

Now, we are ready to define the time series momentum features, which we also refer to as oscillators or individual momentum features interchangeably.

Definition 4.2.6 (Time series momentum features)**.** Given a volatility-scaled price time series for market m from time $t = 1$ to $t = T$, $\tilde{P}_m = (\tilde{P}_{1,m}, \ldots, \tilde{P}_{T,m})$. Given a speed parameter $k \in \mathbb{Z}_+$, we define two smoothing factors as,

$$\alpha_{fast}(k) := \frac{1}{2^k}, \quad \alpha_{slow}(k) := \frac{1}{3 \times 2^k}.$$

Following Definition 4.2.1, we define one exponential weighted moving average of the volatility-scaled price at time t with fast decay and one with slow decay by using the $\alpha_{fast}(k)$ and $\alpha_{slow}(k)$ as a smoothing factor respectively as

$$\mu_{t,m}^{\alpha_{fast}(k)}, \quad \mu_{t,m}^{\alpha_{slow}(k)}.$$

The time series momentum feature with speed k for market m at time t is defined as

$$R_{t,m}^k = \mu_{t,m}^{\alpha_{fast}(k)} - \mu_{t,m}^{\alpha_{slow}(k)}.$$

The underlying intuition is that the crossover of exponentially weighted moving averages can provide insight into recent market trends. If the short-term average price crosses above the long-term average price from below, it indicates an expected increase in price, suggesting a potential upward trend and, conversely, a downward trend if it crosses below. Also, for smaller speed parameters k, the time series momentum feature contains information for short-term recent trends. Conversely, time series momentum features with larger speed parameter k contain long-term trend information. To illustrate this concept, Figure 4.1 demonstrates how the fastest oscillator with speed $k = 1$, captures short-term up and down trends in a more volatile manner. In contrast, a slower oscillator with speed $k = 6$ ultimately exhibits the long-term downtrend.

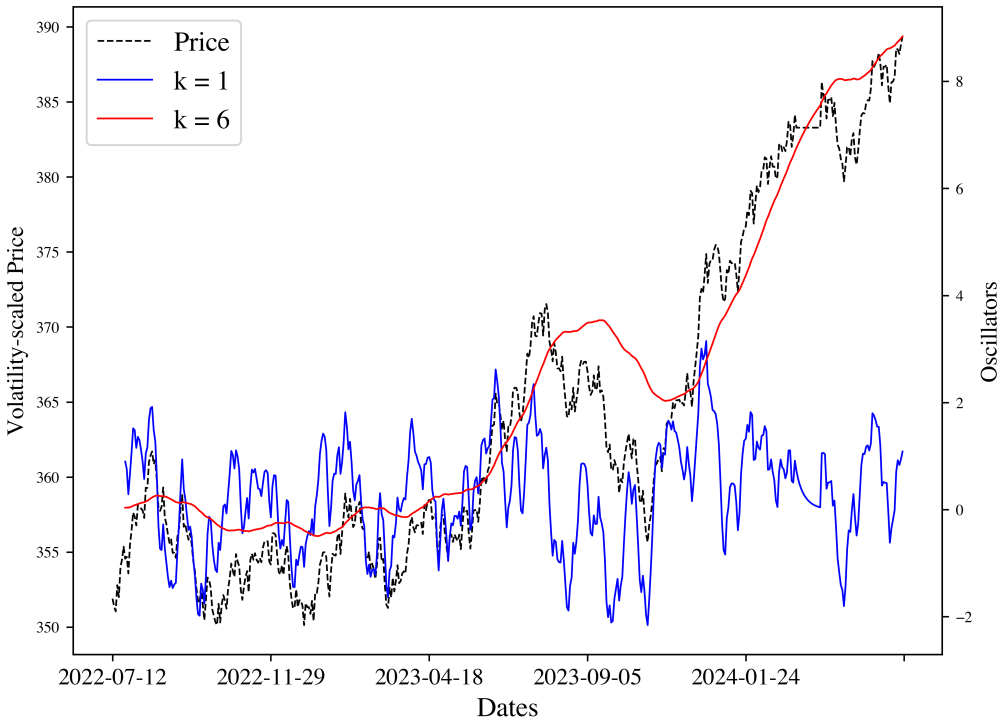


Figure 4.1: Comparison of time series momentum (oscillators) at different speeds for S&P 500 E-mini Futures from July 2022 to June 2024. The fast oscillator ($k = 1$) captures short-term volatility, while the slow oscillator ($k = 6$) highlights the long-term trend.

Throughout this paper, we choose $k = \{1, 2, 3, 4, 5, 6\}$ to create 6 time series momentum features at different speeds to identify the auto-correlation in each market. This is in line with extant literature [7, 56, 40].

4.3 Set Up for Network Momentum Features

In this section, we introduce the setup for the network momentum features. At each trading day t, given a lookback window of δ days, the first step of constructing network momentum features is to use one of the lead-lag detection algorithms to construct a lead-lag matrix $\mathbf{V}_t$ by using the volatility-scaled prices deltas $\tilde{\mathbf{\Delta}}_t \in \mathbb{R}^{\delta \times M}$ as input features. Each vector in $\tilde{\mathbf{\Delta}}_t$ is the volatility-scaled price delta for a market across the past δ days from day t. The candidate algorithms include the Lévy-area algorithm in Table 1.1, one-dimensional dynamic time warping algorithms in Table 2.2, and the multi-dimensional dynamic time warping algorithms in Table 2.3.

The second step is to apply the graph learning algorithm in Table 3.1 by using the lead-lag matrix $\mathbf{V}_t$ as an input feature to obtain the normalised adjacency matrix $\tilde{\mathbf{A}}_t$.

We are now ready to define the network momentum feature.

Definition 4.3.1 (Network momentum feature)**.** Given a time series momentum feature with speed k for market m at time t, $R^k_{t,m}$, and the normalised adjacency matrix $\tilde{\mathbf{A}}_t$ from fitting a lead-lag detection model and the graph learning model to the volatility-scaled price deltas $\tilde{\mathbf{\Delta}}_t$, the network momentum feature with speed k for market m at time t is defined as

$$\tilde{R}^k_{t,m} := \sum_{n \in \mathcal{N}_t(m)} \tilde{A}_{m,n} R^k_{t,n},$$

where $\mathcal{N}_t(m)$ denotes the set of markets connected to market m such that $\tilde{A}_{m,n} \neq 0$ and $R^k_{t,n}$ is the time series momentum feature with the same speed for market n at time t.

Considering that the graph sparsity is significantly influenced by the two hyperparameters α and β in the graph learning model 3.2.1, which consequently affect the number of connections each market can establish, we conduct a discrete grid search over the combinations of:

$$\alpha = \{0.001, 0.01, 0.1, 1, 10, 100\}, \quad \beta = \{0.001, 0.01, 0.1, 1, 10, 100\},$$

on in-sample data to determine their optimal combination for achieving the highest net Sharpe ratio.

We choose $\delta = 132$, so the lead-lag matrix is constructed by considering each market's past half year's daily performances. In addition, to enhance the robustness of our model, we consider employing an ensemble method that fits multiple lead-lag matrices to $\tilde{\mathbf{\Delta}}_t$ across a range of lookback windows. Specifically, we use the following series of lookback windows:

$$\delta = \{22, 44, 66, 88, 110, 132\}.$$

The multiple lead-lag matrices are summarised into a series, which serves as a new input to the graph learning algorithm detailed in Table 3.1. According to [56], employing an ensemble method helps reduce the variance of the learned edge weights, improving the strategy's performance and reducing turnover.

4.4 Portfolio Construction

In contrast to existing literature [72, 63, 52] in which they construct discrete and binary position signals of ± 1 based on the sign of momentum features, we introduce an alternative

approach in this section. Our approach generates a continuous position signal, which has been shown to preserve convexity and positive skewness in returns in [45].

We introduce a scaling function proposed in [45], which we refer to as the response function.

Definition 4.4.1 (Response function)**.** The response function $r(x) : \mathbb{R} \to \mathbb{R}$, parameterised by a positive constant $\lambda > 0$, is defined as follows:

$$r(x) := c_\lambda \cdot x \cdot e^{-\lambda^2 x^2/2},$$

where the normalisation constant c_λ is given by:

$$c_\lambda = (1 + 2\lambda^2)^{3/4}.$$

We pass the obtained momentum signals through the scaling function $r(x)$ to temper excessively high momentum to obtain a moderated momentum signal, theoretically reducing reversal risks by reducing the signal at high trend strengths. The function peaks at $x = \pm\lambda^{-1}$, setting the bounds for maximum positional exposure. Such scaling reduces the volatility of the momentum features, and this is compensated by the normalisation constant c_λ; we refer interested readers to [45] for details on the property of the response function. We fix the parameter $\lambda = \sqrt{2}$.

Definition 4.4.2 (Position signal)**.** Given a series of momentum features with different speeds for market m at time t, $(R^1_{t,m}, \ldots, R^K_{t,m})$, the position signal for market m at time t is defined as

$$X_{t,m} := \left(\frac{1}{M} \frac{1}{K} \sum_{k=1}^{K} r(R^k_{t,m}) \right) \cdot (F_{t,m} \cdot E_{t,m} \cdot \sigma^{22}_{t,m})^{-1} \cdot \Gamma \cdot \frac{\sigma_{\text{tgt}}}{\sqrt{252}},$$

where

- $F_{t,m} \in \mathbb{R}$ denotes the number of units traded in each futures contract for market m on day t.
- $E_{t,m} \in \mathbb{R}$ denotes the exchange rate between the currency in which market m trades and the USD on day t.
- $\sigma^{22}_{t,m}$ is the exponential weighted moving standard deviation of the price delta of market m at time t over a span of 22 days.
- $\Gamma \in \mathbb{R}$ denotes the total capital invested in the strategy, which is always referred to as the asset under management.
- $\sigma_{\text{tgt}} \in \mathbb{R}$ denotes the annual target portfolio volatility. We fix it at 10%.
- $r(\cdot)$ is the response function in Definition 4.4.1.

In this construction, we take equal contributions from oscillators with different speeds and markets in our portfolio, accounted for by the scalars $\frac{1}{M}$ and $\frac{1}{K}$ respectively. The term $(F_{t,m} \cdot E_{t,m} \cdot \sigma^{22}_{t,m})^{-1}$ is the number of contracts for market m on day t required to achieve 1 USD of risk. The last part in the above definition, $\left(\Gamma \cdot \frac{\sigma_{\text{tgt}}}{\sqrt{252}}\right)$, is used to scale our position to realise the daily USD risk amount that our portfolio is targeting.

The position signal for network momentum strategies for market m at time t, $\tilde{X}_{t,m}$, is defined similarly by replacing the momentum features in Definition 4.4.2 with the network momentum features in Definition 4.3.1.

The daily gross return from market m generated by the position signal $X_{t,m}$ for time series momentum features is calculated as

$$r_{t+2,m} := X_{t,m} \cdot \Delta_{t+2,m} \cdot F_{t+2,m} \cdot E_{t+2,m}.$$

The daily gross return for network momentum features $\tilde{r}_{t+2,m}$ is calculated similarly by replacing $X_{t,m}$ with $\tilde{X}_{t,m}$.

We notice that returns from the position signal $X_{t,m}$ can only be realised on day $t+2$, in contrast to approaches in [56, 13, 72] that attribute returns to day $t+1$. This adjustment is necessary because the position signal for day t is generated using the adjacency matrix $\tilde{\mathbf{A}}_t$, which incorporates the closing price of day t; the momentum features defined in Definitions 4.2.6 and 4.3.1 also utilise the closing price of day t for calculating the exponential weighted moving average. Consequently, trades can only be executed at the start of day $t+1$, allowing positions to be established by the end of day $t+1$ and returns to be realised from price changes occurring from day $t+1$ to day $t+2$. This conservative approach reflects the strategy's profitability more accurately in a realistic trading environment.

Given the spread between the bid-ask at closing on day t for market m as $s_{t,m}$, the transaction cost for establishing the position $X_{t,m}$ on day $t+1$ is calculated as

$$c_{t+1,m} := |X_{t+1,m} - X_{t,m}| \cdot \frac{s_{t+1,m}}{2} \cdot F_{t+1,m} \cdot E_{t+1,m}.$$

Here we estimate the cost for executing the trade by half of the spread $s_{t,m}$, and this execution happens during day $t+1$. We denote the transaction cost for network momentum features as $\tilde{c}_{t+1,m}$ by replacing $X_{t,m}$ with $\tilde{X}_{t,m}$.

In the end, we can calculate the net return for market m on time t by

$$r'_{t,m} = r_{t,m} - c_{t,m}.$$

This formulation means that the net return generated by market m on day t consists of the gross return realised from position $X_{t-2,m}$ combined with the transaction cost incurred for establishing the new position generated by $X_{t-1,m}$. The net return for network momentum features is denoted as $\tilde{r}'_{t,m}$ by replacing $X_{t,m}$ with $\tilde{X}_{t,m}$.

We utilise a uniform methodology for portfolio construction and returns calculation across a series of candidate network momentum models, each employing different lead-lag detection algorithms. The models and their configurations are defined as follows:

1. **MACD** uses the time series momentum signal R^k_m, for k from 1 to 6, as defined in Definition 4.2.6, to calculate the position signal in Definition 4.4.2. This model serves as our benchmark.

2. **NMM-DTW** and **NMM-DTW-E**:

   - At each training time t, NMM-DTW constructs the lead-lag matrix $\mathbf{V}_t$ using the classical dynamic time warping algorithm from Table 2.2 with a $\delta = 132$ lookback window. This matrix inputs into the graph learning model and computes the normalised adjacency matrix $\tilde{\mathbf{A}}_t$ following the algorithm in Table 3.1. The network momentum features derived from $\tilde{\mathbf{A}}_t$ following Definition 4.3.1 are then used to calculate the position signal (Definition 4.4.2).
   - NMM-DTW-E employs ensemble methods by fitting the DTW algorithm to varying lookback windows $\delta = \{22, 44, 66, 88, 110, 132\}$. The resulting series of lead-lag matrices serves as the new input to the graph learning model. Subsequent steps follow the NMM-DTW process.

3. **NMM-DDTW** and **NMM-DDTW-E**:

- NMM-DDTW uses the derivative dynamic time warping algorithm from Table 2.2. Subsequent steps follow the NMM-DTW process.
- NMM-DDTW-E employs the ensemble methods by constructing the lead-lag matrices with DDTW algorithm with lookback $\delta = \{22, 44, 66, 88, 110, 132\}$. Subsequent steps follow the NMM-DTW-E process.

4. **NMM-SDTW** and **NMM-SDTW-E**:
   - NMM-SDTW applies the shape dynamic time warping algorithm with the descriptor length $l = 11$ from Table 2.3 to construct the lead-lag matrix with a lookback window of $\delta = 132$. Subsequent steps follow the NMM-DTW process.
   - NMM-SDTW-E employs the ensemble methods with shape dynamic time warping with lookback windows $\delta = \{22, 44, 66, 88, 110, 132\}$. Subsequent steps follow the NMM-DTW-E process.
5. **NMM-SDDTW** and **NMM-SDDTW-E** :
   - NMM-SDDTW applies the shape dynamic time warping derivative algorithms with the descriptor length $l = 11$ from Table 2.3 to construct the lead-lag matrix with a lookback window of $\delta = 132$. Subsequent steps follow the NMM-DTW process.
   - NMM-SDTW-E employs the ensemble methods with lookback windows $\delta = \{22, 44, 66, 88, 110, 132\}$. Subsequent steps follow the NMM-DTW-E process.
6. **NMM-LEVY** and **NMM-LEVY-E**:
   - NMM-LEVY applies the Lévy area algorithm in Table 1.1 to construct the lead-lag matrix with lookback window $\delta = 132$. Subsequent steps follow the NMM-DTW process.
   - NMM-LEVY-E employs the ensemble methods with Lévy area algrotim with lookback windows $\delta = \{22, 44, 66, 88, 110, 132\}$. Subsequent steps follow the NMM-DTW-E process.

4.5 Performance Analysis

4.5.1 Portfolio Performance Analysis

In evaluating model performance, we consider the following three aspects, following the convention established in [56]:

1. **Profitability**: This includes annualised expected gross return, annualised expected net return, and hit rate – defined as the percentage of days with positive returns during the out-of-sample periods.
2. **Risk**: This includes volatility, downside deviation, and maximum drawdown to understand the risk exposure of our models. While comparing the volatility and downside deviation across models is unnecessary due to each model's positions being scaled to a target volatility of 10% (Definition 4.4.2), we include them in our summary for completeness, as they remain relevant for calculating the Sharpe ratio and Sortino ratio.
3. **Overall and Other Performance**: This includes transaction costs, skewness of monthly returns, Sharpe ratio (expected return / volatility), Sortino ratio (expected return / downside deviation), Calmar ratio (expected return / maximum drawdown), and the ratio of average profits to average losses $\left(\frac{\text{Avg. P}}{\text{Avg. L}}\right)$.

We present the performance of the benchmark MACD model alongside the network momentum models. Our primary focus is on their average performance across 100 bootstrapped datasets. We investigate whether the network momentum models can achieve a statistically significant higher net Sharpe ratio compared to the benchmark MACD model. We also present the models' performance on real-world price data from the out-of-sample period of 2005 to 2024 to illustrate their profitability in a historical context.

Table 4.1: Performance Metrics for Various Signals

	Gross Return	Transaction	Net Return	vol.	Sharpe	downside deviation	MDD	Sortino	Calmar	Skewness	hit rate	Avg. P Avg. L
Panel A: Average Performance on 100 Bootstrapped Price Data												
MACD	0.057	0.027	0.030	0.107	0.277	0.058	0.239	0.515	0.039	0.395	0.516	1.158
NMM-DTW	0.064	0.029	0.034	0.108	0.315	0.058	0.261	0.592	0.041	0.441	0.515	1.200
NMM-DTW-E	0.063	0.023	0.039	0.109	0.353	0.058	0.259	0.669	0.046	0.457	0.516	1.226
NMM-DDTW	**0.064**	0.029	0.034	0.109	0.315	0.059	0.256	0.590	0.042	0.450	0.514	1.203
NMM-DDTW-E	0.063	0.023	**0.039**	0.110	**0.357**	0.058	0.250	**0.684**	0.048	0.486	0.514	1.243
NMM-SDTW	0.064	0.028	0.035	0.110	0.319	0.058	0.280	0.606	0.041	0.458	0.508	1.235
NMM-SDTW-E	0.062	**0.022**	0.039	0.109	0.355	0.058	0.255	0.677	0.047	0.473	0.512	1.250
NMM-SDDTW	0.062	0.029	0.032	0.109	0.296	0.057	0.269	0.568	0.039	0.507	0.504	1.234
NMM-SDDTW-E	0.062	0.023	0.038	0.110	0.350	0.057	0.257	0.679	0.046	**0.509**	0.510	**1.255**
NMM-LEVY	0.064	0.027	0.036	0.109	0.336	0.059	**0.230**	0.624	**0.050**	0.419	**0.517**	1.206
NMM-LEVY-E	0.060	0.024	0.035	0.109	0.323	0.058	0.240	0.610	0.045	0.454	0.516	1.202
Panel B: Performance on Real Price Data												
MACD	0.051	0.026	0.024	0.104	0.233	0.053	0.227	0.454	0.031	0.645	**0.526**	1.080
NMM-DTW	0.054	0.028	0.026	0.106	0.243	0.056	0.274	0.457	0.027	0.630	0.513	1.147
NMM-DTW-E	0.062	0.023	**0.039**	0.106	**0.364**	0.056	0.203	0.694	0.055	0.683	0.509	1.285
NMM-DDTW	0.056	0.028	0.027	0.107	0.257	0.053	0.247	0.517	0.032	**0.759**	0.487	1.282
NMM-DDTW-E	0.055	0.022	0.032	0.107	0.298	0.055	0.244	0.577	0.038	0.719	0.513	1.198
NMM-SDTW	**0.065**	0.027	0.037	0.106	0.351	0.054	**0.163**	0.689	**0.066**	0.704	0.513	1.249
NMM-SDTW-E	0.055	0.022	0.033	0.107	0.307	0.055	0.222	0.600	0.043	0.691	0.513	1.205
NMM-SDDTW	0.049	0.028	0.020	0.108	0.189	0.058	0.289	0.354	0.020	0.589	0.470	**1.303**
NMM-SDDTW-E	0.057	**0.022**	0.035	0.106	0.328	0.054	0.208	0.643	0.048	0.720	0.509	1.249
NMM-LEVY	0.064	0.026	0.038	0.106	0.356	0.054	0.204	**0.702**	0.053	0.675	0.517	1.228
NMM-LEVY-E	0.055	0.023	0.032	0.106	0.300	0.054	0.208	0.586	0.044	0.673	0.513	1.198

[a] Best performance is in bold. [b] No comparison for volatility and downside deviation since every portfolio is scaled to the same target annualised volatility in 4.4.2 for direct comparison of the net Sharpe.

In Panel A of Table 4.1, we report the average performance of the portfolio constructed from various momentum models on bootstrapped price data. In Panel B, we report the performance of these models on real price data from 2005 to 2024.

Based on the metrics in Panel A, all network momentum models (NMM) exhibit higher expected gross returns than the benchmark MACD model, with the NMM-DDTW model achieving the highest at 0.064, compared to MACD's 0.057. Typically, NMM models incur higher transaction costs than MACD, reflecting their sensitivity to market movements and increased daily turnover. However, ensemble methods reduce transaction costs, with DTW variations further decreasing them to 0.022, approximately 19% lower than MACD. The NMM-LEVY model achieves an 11% reduction in costs. As a result, all NMM models demonstrate better performance over MACD in terms of expected net returns, net Sharpe ratios, and Sortino ratios. Notably, NMM-DDTW-E achieves a Sharpe ratio of 0.357 and a Sortino ratio of 0.684, marking increases of 29% and 33%, respectively, over MACD.

The ability to effectively follow trends is crucial for trading strategies. NMM-SDDTW-E stands out with the highest $\frac{\text{Avg. Profit}}{\text{Avg. Loss}}$ ratio among all NMM models. It also exhibits the highest positive skewness, suggesting that although it may frequently incur small losses, the gains it captures are significant. Meanwhile, NMM-LEVY demonstrates the smallest MDD and highest hit rate, suggesting it is particularly effective at identifying trend reversals and capturing new opportunities for positive returns, although it achieves smaller gains per trade, as indicated by its slightly lower skewness and the ratio between average profit and average loss.

Panel B of Table 4.1 demonstrates that NMM models outperform the benchmark MACD model on real market data during the out-of-sample period from 2005 to 2024. NMM-DTW-E achieves the highest net Sharpe ratio at 0.364, compared to the benchmark's 0.233, showing better risk-adjusted returns. NMM-DDTW exhibits the most pos-

itive skew in returns at 0.759, surpassing the benchmark's 0.645. Although MACD has the highest hit rate, indicating more days with positive PnL, it suffers from the lowest $\frac{\text{Avg. Profit}}{\text{Avg. Loss}}$ ratio, suggesting that its losses are larger than those of NMM models. However, we reiterate that the portfolio included in Appendix B.1 is somewhat random, and the models' performance may not be reproducible for other portfolios. Therefore, we emphasise that our assessment of the models is primarily based on their performance on the bootstrapped data.

We examine the distribution of the net Sharpe ratios for all models. Figure 4.2 presents the distribution of Sharpe ratios on the bootstrapped price datasets along with their interquartile ranges. The net Sharpe ratios achieved by each model on the actual price data are marked by red crosses on the distribution plots. The box plots demonstrate that the median net Sharpe ratios for all network momentum models are higher than those for the MACD model, and the ensemble methods further enhance performance. The positioning of the red crosses, which for all models except NMM-DDTW-E and NMM-SDTW-D fall within the interquartile ranges, suggests that the bootstrapped price data provides a valid representation of the real price data and is suitable for inference.

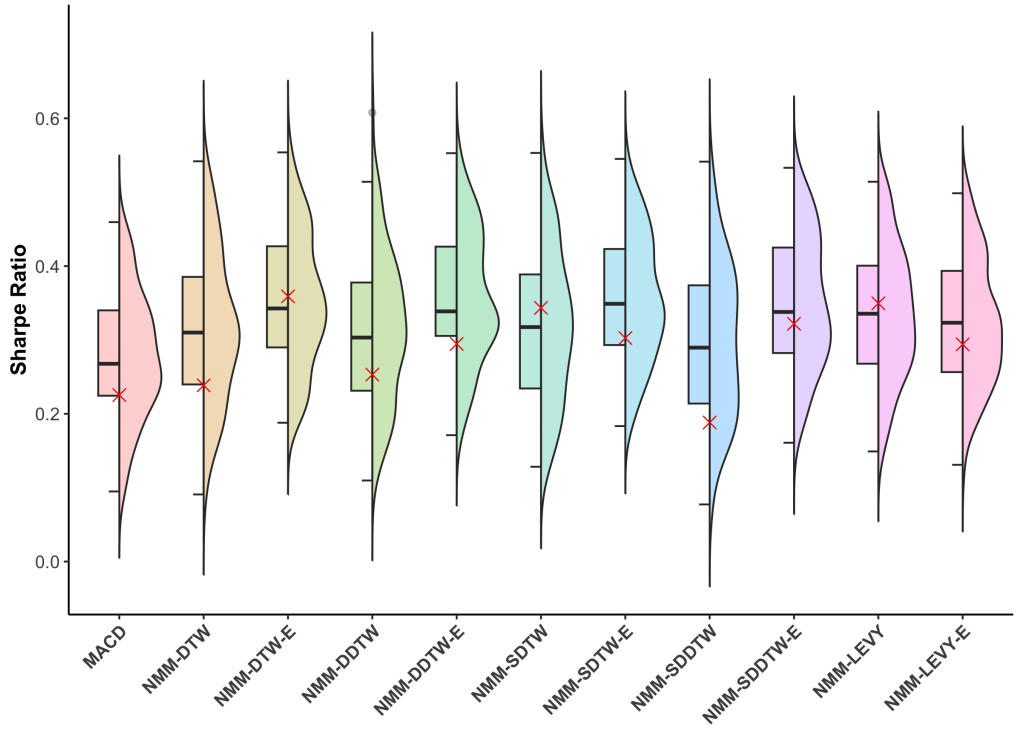


Figure 4.2: Distribution of net Sharpe Ratios for the Benchmark Model (MACD) and Network Momentum Models on bootstrapped datasets, with net Sharpe achieved on real price data indicated by red crosses

We have two primary objectives as follows:

1. To determine whether the net Sharpe ratio achieved by the network momentum model is significantly higher than that achieved by the MACD model when both are used to construct portfolios from the same price data set. We employ a one-sided Wilcoxon signed-rank test [68], a matched-pair test, to assess if the difference in net Sharpe ratios (network momentum model minus MACD model) is significantly greater than 0.

2. To examine whether the distributions of the net Sharpe ratios from the MACD

model and a network momentum model are statistically different without considering the matched-pair nature of the data. We use the one-sided Kolmogorov-Smirnov test [9] to determine if the cumulative distribution function of the MACD model's net Sharpe ratios is stochastically greater than that of the network momentum model, it indicates that the MACD model generally yields lower Sharpe ratios than the network momentum model.

We report the p-values for the two tests in Table 4.2. For the Wilcoxon signed-rank test, all network momentum models achieve significant p-values ($p < 0.05$). This demonstrates that, when applied to the same random set of market price data, the network momentum models significantly outperform the benchmark MACD model, which relies only on time-series momentum in terms of net Sharpe ratio. For the Kolmogorov-Smirnov test, aside from NMM-SDDTW, all other NMM models achieve significant p-values ($p < 0.05$). This indicates that the cumulative distribution function of the net Sharpe ratios for the network momentum models is stochastically smaller than that of the MACD model, suggesting that the network momentum models generally achieve higher Sharpe ratios than the MACD model. These two tests collectively underscore the enhanced performance capability of the network momentum feature.

Our results demonstrate the robustness and reliability of the network momentum spillover identified by the proposed algorithms. These findings suggest that under both uniform and varied market conditions, the NMM models consistently outperform the benchmark MACD model with statistically confidence.

Table 4.2: P-Values for Sharpe Ratio Comparisons Against Benchmark

	NMM-DTW	NMM-DTW-E	NMM-DDTW	NMM-DDTW-E	NMM-SDTW	NMM-SDTW-E	NMM-SDDTW	NMM-SDDTW-E	NMM-LEVY	NMM-LEVY-E
Wilcoxon signed-rank test	0	0	0	0	0	0	0.005	0	0	0
Kolmogorov–Smirnov test	0.018	0	0.012	0	0.005	0	0.077	0	0	0.002

4.5.2 Long/Short Performance Analysis

In this section, We focus on the model's ability to identify and respond to upward and downward market trends by examining performance in both long and short trading positions. The returns from these positions are analysed separately, with the metrics for short and long positions detailed in Tables 4.3 and 4.4, respectively.

Based on the data in Panel A of Table 4.3, the benchmark model MACD averages a loss in short positions on the bootstrapped dataset, with a net Sharpe of -0.396 and the highest MDD across both bootstrapped and real price data. In contrast, NMM models improve performance in short positions by reducing losses. Specifically, NMM-DDTW-E enhances performance over MACD by reducing losses by 35% and increasing the net Sharpe ratio by 24% on bootstrapped data. It also achieves the highest Sortino and Calmar ratios, indicating effective downside risk and MDD control. Despite MACD's higher hit rate in short positions, its skewness score of 0.804 is lower than that of NMM-DDTW-E, which scores 1.155, and other NMM models. This indicates that NMM models not only result in smaller losses but also achieve more substantial occasional gains.

In Panel B of Table 4.3, NMM models continue to demonstrate effective loss control in short positions on the real price data. NMM-SDTW-E and NMM-DTW-E notably improve net Sharpe and reduce MDD to the greatest extent compared to the benchmark, respectively, with NMM-DDTW-E again achieving the most positively skewed performance, mirroring its success on bootstrapped datasets.

In the long direction, as detailed in Table 4.4, MACD demonstrates strong profitability with a net Sharpe ratio of 0.559 with the highest hit rate at 0.554. Among the network momentum models, NMM-LEVY outperforms with a net Sharpe of 0.587, a 6.1% increase over the benchmark. It also reduces the MDD to 0.168, indicating superior loss control.

Table 4.3: Performance Metrics for Various Signals in Short Direction Only

	Gross Return	Transaction	Net Return	vol.	Sharpe	downside deviation	MDD	Sortino	Calmar	Skewness	hit rate	Avg. P Avg. L
Panel A: Average Performance on 100 Bootstrapped Price Data												
MACD	-0.011	0.014	-0.026	0.066	-0.396	0.040	0.635	-0.638	-0.012	0.804	**0.367**	1.245
NMM-DTW	**-0.005**	0.014	-0.020	0.062	-0.329	0.039	0.546	-0.513	-0.010	0.885	0.351	1.379
NMM-DTW-E	-0.006	0.012	-0.018	0.058	-0.317	0.037	0.519	-0.490	-0.010	1.007	0.342	1.427
NMM-DDTW	-0.006	0.014	-0.021	0.062	-0.340	0.038	0.553	-0.541	-0.011	1.042	0.349	1.385
NMM-DDTW-E	-0.005	0.011	**-0.017**	0.059	**-0.300**	0.037	0.508	**-0.467**	**-0.010**	**1.155**	0.341	1.449
NMM-SDTW	-0.007	0.014	-0.022	0.061	-0.364	0.038	0.562	-0.569	-0.011	0.996	0.348	1.358
NMM-SDTW-E	-0.006	0.011	-0.017	0.058	-0.310	0.037	**0.500**	-0.474	-0.010	1.035	0.342	1.439
NMM-SDDTW	-0.007	0.014	-0.021	0.061	-0.351	0.039	0.555	-0.544	-0.011	0.877	0.344	1.395
NMM-SDDTW-E	-0.007	**0.011**	-0.018	0.058	-0.317	0.037	0.510	-0.487	-0.010	1.130	0.333	**1.468**
NMM-LEVY	-0.008	0.014	-0.022	0.062	-0.363	0.039	0.573	-0.576	-0.011	0.814	0.357	1.327
NMM-LEVY-E	-0.010	0.012	-0.022	0.060	-0.374	0.038	0.575	-0.586	-0.011	0.861	0.341	1.387
Panel B: Performance on Real Price Data												
MACD	-0.013	0.014	-0.028	0.070	-0.396	0.043	0.584	-0.645	-0.014	0.953	**0.376**	1.189
NMM-DTW	-0.007	0.014	-0.022	0.066	-0.327	0.040	0.525	-0.536	-0.012	1.218	0.342	1.430
NMM-DTW-E	**-0.005**	0.012	**-0.016**	0.064	-0.254	0.039	**0.450**	**-0.410**	**-0.010**	1.237	0.363	1.368
NMM-DDTW	-0.009	0.015	-0.023	0.066	-0.353	0.040	0.514	-0.578	-0.013	1.295	0.368	1.252
NMM-DDTW-E	-0.007	0.012	-0.019	0.063	-0.299	0.039	0.487	-0.484	-0.011	**1.395**	0.333	1.485
NMM-SDTW	-0.006	0.014	-0.020	0.067	-0.297	0.041	0.456	-0.483	-0.013	1.243	0.350	1.393
NMM-SDTW-E	-0.007	0.012	-0.018	0.062	**-0.297**	0.040	0.485	-0.465	-0.011	1.290	0.355	1.350
NMM-SDDTW	-0.012	0.014	-0.025	0.067	-0.381	0.041	0.553	-0.615	-0.013	1.164	0.312	**1.575**
NMM-SDDTW-E	-0.008	**0.012**	-0.019	0.062	-0.299	0.040	0.485	-0.469	-0.011	1.365	0.329	1.485
NMM-LEVY	-0.006	0.014	-0.020	0.067	-0.296	0.041	0.476	-0.487	-0.012	1.027	0.372	1.312
NMM-LEVY-E	-0.012	0.012	-0.024	0.065	-0.374	0.041	0.549	-0.595	-0.013	0.906	0.359	1.267

[a] Best performance is in bold. [b] No comparison for volatility and downside deviation since every portfolio is scaled to the same target annualised volatility in 4.4.2 for direct comparison of the net Sharpe.

Notably, while some network momentum models exhibit slightly lower net Sharpe ratios in long positions compared to the benchmark, all of them demonstrate more positively skewed returns, signifying smaller average losses and occasional larger gains. NMM-SDDTW achieves the most positively skewed returns, with a 76.6% increase over MACD's skewness. This highlights the robust capability of network momentum models in long positions. Corresponding performance on the real price data in Panel B of Table 4.4 further supports this, showing that NMM-LEVY has a higher Sharpe and Sortino ratio compared to the benchmark, and NMM-SDDTW-E records the most skewed returns.

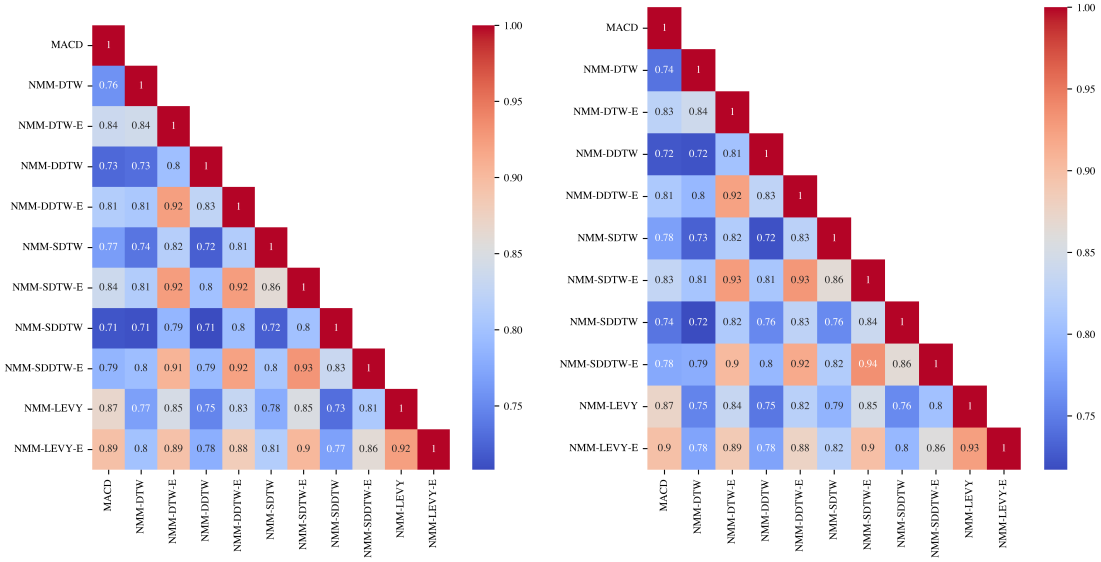
Table 4.4: Performance Metrics for Various Signals in Long Direction Only

	Gross Return	Transaction	Net Return	vol.	Sharpe	downside deviation	MDD	Sortino	Calmar	Skewness	hit rate	Avg. P Avg. L
Panel A: Average Performance on 100 Bootstrapped Price Data												
MACD	0.068	0.012	0.055	0.099	0.559	0.057	0.191	0.983	0.091	0.367	**0.554**	1.243
NMM-DTW	0.069	0.015	0.054	0.100	0.540	0.054	0.186	0.998	0.093	0.553	0.519	1.412
NMM-DTW-E	0.069	0.012	0.057	0.099	0.572	0.054	0.172	1.053	0.103	0.574	0.519	1.451
NMM-DDTW	0.070	0.015	0.055	0.101	0.542	0.055	0.192	1.001	0.090	0.565	0.522	1.401
NMM-DDTW-E	0.068	0.012	0.056	0.099	0.568	0.054	0.173	1.053	0.101	0.594	0.518	1.459
NMM-SDTW	0.071	0.014	0.057	0.101	0.563	0.055	0.188	1.047	0.098	0.557	0.525	1.405
NMM-SDTW-E	0.068	**0.011**	0.056	0.099	0.569	0.054	0.171	1.056	0.102	0.602	0.518	1.460
NMM-SDDTW	0.069	0.015	0.053	0.100	0.529	0.054	0.193	0.997	0.088	**0.648**	0.517	1.417
NMM-SDDTW-E	0.068	0.012	0.056	0.099	0.569	0.053	0.172	1.066	0.102	0.648	0.517	**1.468**
NMM-LEVY	**0.072**	0.013	**0.059**	0.100	**0.587**	0.055	**0.168**	**1.076**	0.109	0.477	0.534	1.377
NMM-LEVY-E	0.069	0.012	0.057	0.098	0.582	0.054	0.161	1.076	**0.110**	0.542	0.527	1.415
Panel B: Performance on Real Price Data												
MACD	0.064	0.012	0.052	0.094	0.557	0.049	0.199	1.065	0.076	0.623	**0.547**	1.276
NMM-DTW	0.062	0.014	0.047	0.096	0.494	0.048	0.216	0.995	0.063	0.901	0.500	1.479
NMM-DTW-E	0.067	0.011	0.055	0.095	0.581	0.047	0.158	1.170	0.100	0.918	0.491	**1.649**
NMM-DDTW	0.065	0.014	0.051	0.098	0.519	0.050	0.214	1.011	0.069	0.831	0.496	1.531
NMM-DDTW-E	0.062	0.011	0.051	0.095	0.535	0.048	0.193	1.052	0.076	0.868	0.517	1.418
NMM-SDTW	0.070	0.013	0.057	0.095	0.600	0.046	**0.153**	1.236	**0.108**	0.916	0.513	1.538
NMM-SDTW-E	0.062	**0.011**	0.051	0.094	0.543	0.047	0.170	1.088	0.087	0.904	0.500	1.545
NMM-SDDTW	0.061	0.014	0.046	0.098	0.470	0.047	0.220	0.972	0.060	0.850	0.500	1.441
NMM-SDDTW-E	0.065	0.011	0.053	0.095	0.565	0.047	0.168	1.139	0.092	**0.935**	0.504	1.550
NMM-LEVY	**0.070**	0.012	**0.058**	0.094	**0.610**	0.046	0.169	**1.257**	0.098	0.810	0.526	1.449
NMM-LEVY-E	0.068	0.011	0.056	0.094	0.597	0.047	0.155	1.187	0.105	0.813	0.517	1.495

[a] Best performance is in bold. [b] No comparison for volatility and downside deviation since every portfolio is scaled to the same target annualised volatility in 4.4.2 for direct comparison of the net Sharpe.

4.5.3 Diversification Analysis

We analyse the correlation of their returns to assess whether the NMM models and MACD exhibit orthogonal trading signals. Figure 4.3(a) presents the average correlation on bootstrapped datasets, while Figure 4.3(b) the correlation on real price data covering the entire out-of-sample period from 2005 to 2024.



(a) (b)

Figure 4.3: A diversification analysis on the PnL pairwise correlation between models on the bootstrapped datasets (left) and real price dataset (right).

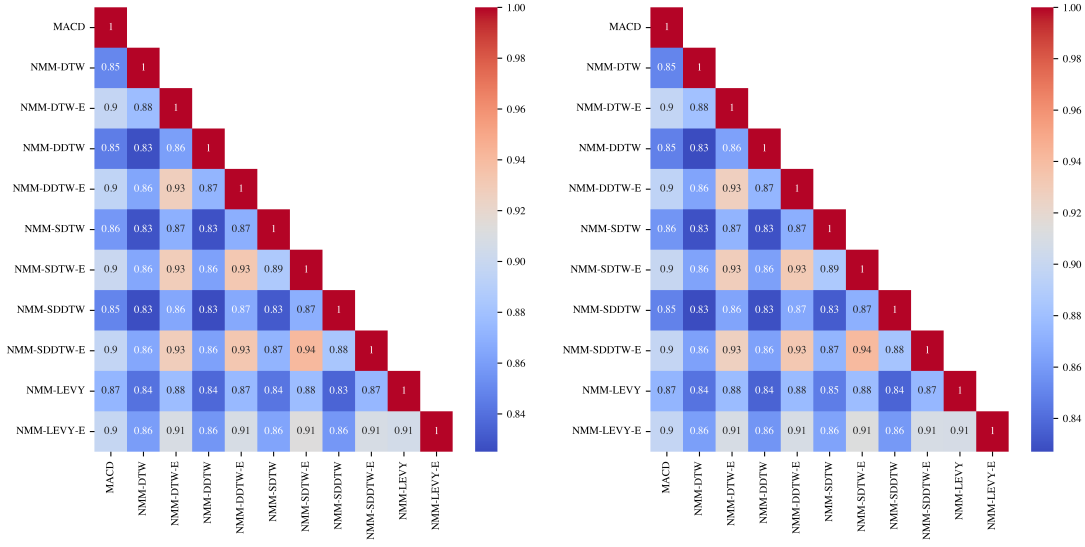
By analysing the returns between the NMM models and the benchmark MACD on the bootstrapped datasets, we notice that the average correlations range from 0.71 to 0.89 in Figure 4.3(a). NMM-SDDTW exhibits the lowest average correlation with MACD at 0.71, similar to NMM-DDTW's correlation with MACD. NMM-LEVY and NMM-LEVY-E show slightly higher correlations with MACD at 0.87 and 0.89, respectively. Comparable results are observed in the PnL from the real price data between 2005 and 2024 in Figure 4.3(b), where NMM-DDTW and NMM-SDDTW display the lowest correlations with MACD, at 0.72 and 0.74, respectively. Although the PnL correlations are not completely orthogonal, these empirical findings support the existence of additional information captured in our NMM models.

Our empirical findings indicate that different DTW algorithms capture distinct lead-lag relationships, consequently influencing the network momentum identified. Specifically, NMM models employing multi-dimensional DTW algorithms, such as NMM-SDTW and NMM-SDDTW, exhibit lower correlation values, around 0.7, with models based on one-dimensional DTW algorithms like NMM-DTW and NMM-DDTW in Figure 4.3(a). This suggests that multi-dimensional DTW effectively captures different lead-lag relationships with one-dimensional approaches. Furthermore, NMM-LEVY demonstrates correlations ranging from 0.73 to 0.83 with NMM-DTW and its variations, indicating that using the Lévy area as a lead-lag detection method yields additional results from those obtained via dynamic time warping algorithms.

It is also noteworthy that correlations between each NMM model and its ensemble variant range from 0.80 to 0.92. This implies that while there is some dependency, the ensemble method still introduces different information on the lead-lag relationship. This is achieved by utilising six different lookback lengths, leading to variations in the net-

work momentum model outcomes. The ensemble approach thus contributes uniquely to understanding and leveraging network momentum in trading strategies.

Next, we introduce a second metric for our diversification analysis: the sign agreement between two models. This metric is the percentage of days on which two models share the same trading direction—either opting to go long or short on the market on a trading day—across the entire portfolio. The average results on bootstrapped data is presented in Figure 4.4(a), with performance on real price data from the entire out-of-sample period from 2005 to 2024 in Figure 4.4(b). We also examine the average annualised expected PnL on days when the NMM models diverge in sign from the benchmark MACD model to assess whether differences in trading direction result in additional profits. The differences in average profits between the models (network momentum models minus MACD) for these days are detailed in Table 4.5.



(a) (b)

Figure 4.4: A diversification analysis on the pairwise sign agreement between models on the bootstrapped datasets (left) and real price dataset(right)

Table 4.5: Average PnL Gains Over Benchmark on Opposing Signal Days

	NMM-DTW	NMM-DTW-E	NMM-DDTW	NMM-DDTW-E	NMM-SDTW	NMM-SDTW-E	NMM-SDDTW	NMM-SDDTW-E	NMM-LEVY	NMM-LEVY-E
Bootstrapped data	0.011	**0.032**	0.001	0.023	-0.002	0.021	-0.020	0.026	0.013	0.004
Real Price data	0.017	0.043	-0.017	0.015	**0.057**	-0.008	-0.063	0.039	0.051	0.031

It can be observed that NMM-DTW and NMM-DDTW, with the lowest sign agreement with the MACD at 85%, result in average additional returns of 0.011 and 0.032, respectively. Most other NMM models show a sign agreement ranging from 85% to 90% with MACD and generally yield higher returns than MACD on days with differing signs, except for NMM-SDTW and NMM-SDDTW, which achieve less profits than the benchmark on these days. These empirical results suggest that the additional network momentum captured by the NMM models is effective at following and adjusting to trends identified by the MACD, which focuses solely on time-series momentum. This indicates that our models are robust and effective in identifying network momentum within a portfolio.

On the real price data from 2005 to 2024, it is notable that the NMM-SDTW achieves the highest average returns gain over the benchmark model with an annualised expected difference at 0.057, with a sign agreement of 86%. However, NMM-DDTW, NMM-SDTW-E, and NMM-SDDTW realise lower profits compared to MACD, with respective losses of

-0.017, -0.008 and -0.063, respectively, , and sign agreements of 85%, 90%, and 85%.

4.5.4 Skewness Analysis

In this final section on performance analysis, we examine the skewness of returns from NMM models across different time horizons and compare them with the benchmark MACD model.

As highlighted in [45], effective trend-following strategies often exhibit a long-option-type payoff, attributed to positive skewness. This phenomenon can be conceptualised as the purchase of an option: regular small losses represent the premium paid, while correctly identifying and riding a trend may result in significant gains, analogous to an option's payoff.

We present the skewness across various return horizons for four NMM models in Figure 4.5 for detailed analysis. The four representative network momentum models are: NMM-DTW-E (a), which performs the best on real price data and achieves the highest average PnL gain over the benchmark on days with opposing signals; NMM-DDTW-E (b), the top performer on bootstrapped data and in short positions; NMM-SDTW-E (c), which outperforms the other multi-dimensional dynamic time warping models; and NMM-LEVY (d), the top performer in long positions. Our empirical study finds that the skewness of the network momentum models exhibits a similar and consistent pattern; therefore, we only include four examples here. For completeness, we include plots for the other NMM models in Appendix B.2.

Our analysis indicates that the NMM models exhibit stronger positive skewness in returns across time horizons ranging from days to months compared to the benchmark MACD, with a notable peak at the two-month return horizon. This suggests that the NMM models are more effective at identifying and positioning for trends ahead of time to capitalise on these opportunities. Even in the case of daily returns, where all models typically exhibit a negative skew due to the option-like payoff characteristic of trend-following strategies, the NMM models show less negative skewness, indicating better risk control and the ability to identify short-term trends without enduring prolonged periods of losses.

However, it is important to note that the NMM models demonstrate a more pronounced decay in skewness over longer horizons compared to MACD. Particularly from half-year to one-year return horizons, although the NMM models still maintain positive skewness, it is less pronounced than that observed with MACD.

The pattern of skewness across different time horizons aligns with the findings reported in [45], which we refer interested readers to for further details. Our empirical results suggest that NMM models not only uphold the desired characteristics of a trend-following strategy but also enhance them. They effectively capture network momentum spillover and identify both short-term and medium-term trends accurately, thereby enabling the models to anticipate market movements by considering momentum from interconnected markets within the portfolio.

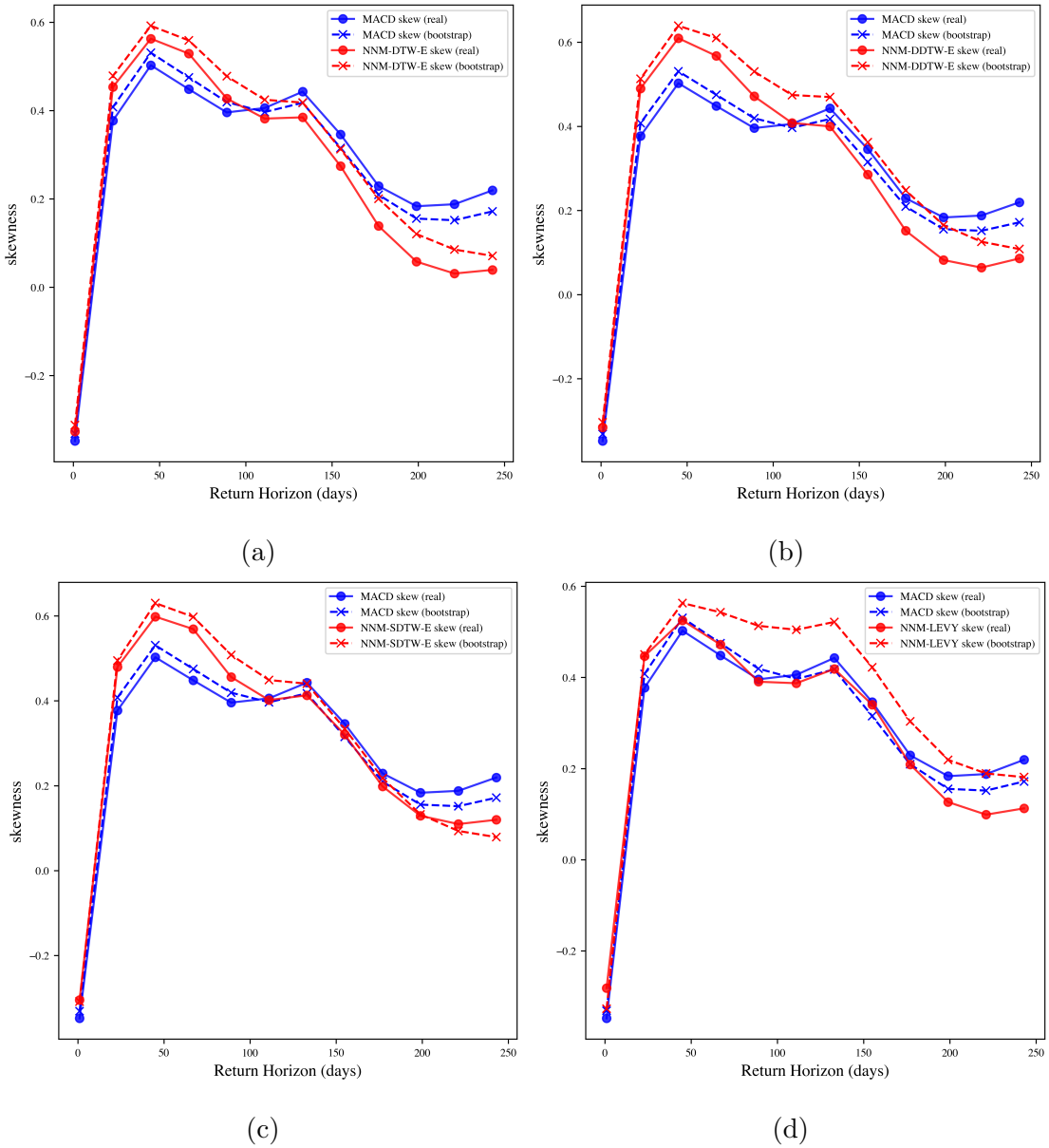


Figure 4.5: Skewness in the returns of the network momentum model over various periods, compared to those of the time series momentum model, using different lead-lag detection models: (a) NMM-DTW-E, (b) NMM-DDTW-E, (c) NMM-SDTW-E, and (d) NMM-LEVY.

Conclusion

We propose a methodology that transforms cross-sectional momentum spillover into network momentum across market industries. This process utilises two lead-lag detection models to identify non-linear relationships at fixed lags and between non-synchronised market returns. We then apply a graph learning model to quantify the intricate interconnectedness of market leadership and individual momentum, generating a novel trading signal. This signal is utilised to construct a portfolio for a systematic trend-following strategy, which we evaluate using 100 sets of bootstrapped price data from 28 futures contracts across metals, agriculture, energy, and equities. We backtest our strategy in a realistic trading environment that accounts for time delays in establishing positions.

Our framework enhances the performance of traditional trend-following strategies, consistently achieving a higher and statistically significant net Sharpe ratio compared to time series momentum strategies. Our model also robustly reduces transaction costs and enhances performance over time series momentum strategies in short positions, where the latter typically incurs losses. By employing various lead-lag detection techniques, our network momentum models generate low-correlated signals that more effectively identify market trends by establishing positions in the correct direction. The proposed framework also consistently yields more positively skewed returns, underscoring the efficiency and robustness of the network momentum identified for trend-following strategies.

Most importantly, the results indicate that the superior performance of converting cross-sectional momentum into network momentum is not confined to specific market combinations within the portfolio, nor is it dependent on historical market trends. Instead, the proposed network momentum model demonstrates remarkable generalisability across various industries and markets.

We propose several future research directions. Firstly, exploring non-linear ensemble methods on the lead-lag matrices computed by multiple models could be beneficial. Considering that the divergence analysis indicates dynamic time warping and Lévy area models capture different information, their combination in a non-linear manner may enhance the identification of lead-lag relationships. Secondly, investigating asymmetrical adjacency matrices with machine learning models like graph neural networks could shed light on potential non-symmetrical relationships between markets. Thirdly, while our current portfolio construction combines time series momentum features with equal weights and applies the same adjacency matrix to all of them, it may be worthwhile to explore fitting different lead-lag matrices and adjacency matrices to time series momentum features at varying speeds. Employing non-linear methods to combine these may more effectively capture the nonlinearity in momentum spillover.

Appendix A

Technical Proof

A.1 Derivation of the Formula for Lévy Area Between Discrete Processes

In this section, we provide the derivation of the Lévy area between two discrete processes in Formula (1.2.1).
Consider a time interval $t \in [a, b]$, and let $X^i = \{X^i_a, \dots, X^i_b\}$ and $X^j = \{X^j_a, \dots, X^j_b\}$ be two discrete stochastic processes. The Lévy area, as defined in 1.1.4, is given by:

$$A^{\text{Lévy}} = \frac{1}{2}(S(X)^{i,j}_{a,b} - S(X)^{j,i}_{a,b}). \tag{A.1.1}$$

This equation encompasses the difference in the cross-terms of the double-iterated integrals for the two processes, expressed as:

$$\begin{aligned} S(X)^{i,j}_{a,b} - S(X)^{j,i}_{a,b} &= \int_{a<s<b}\int_{a<t<s} dX^i_t dX^j_s - \int_{a<s<b}\int_{a<t<s} dX^j_t dX^i_s \\ &= \int_{a<s<b} (X^i_s - X^i_a) dX^j_s - \int_{a<s<b} (X^j_s - X^j_a) dX^i_s. \end{aligned}$$

This integration translates into finite summations:

$$\begin{aligned} S(X)^{i,j}_{a,b} - S(X)^{j,i}_{a,b} &= \sum_{a<s<b} (X^i_s - X^i_a)\Delta X^j_s - \sum_{a<s<b} (X^j_s - X^j_a)\Delta X^i_s \\ &= \sum_{a<s<b} (X^i_s - X^i_a)(X^j_s - X^j_{s-1}) - \sum_{a<s<b} (X^j_s - X^j_a)(X^i_s - X^i_{s-1}) \\ &= \sum_{a<s<b} (-X^i_s X^j_{s-1} + X^j_s X^i_{s-1}) + \sum_{a<s<b} (X^i_a(X^j_{s-1} - X^j_s) \\ &\quad + X^j_a(X^i_s - X^i_{s-1})). \end{aligned}$$

These summations can be simplified to:

$$S(X)^{i,j}_{a,b} - S(X)^{j,i}_{a,b} = \sum_{a<s<b} (-X^i_s X^j_{s-1} + X^j_s X^i_{s-1}) + X^i_a(X^j_a - X^j_b) + X^j_a(X^i_b - X^i_a). \tag{A.1.2}$$

By substituting (A.1.2) into (A.1.1), we achieve the desired formula (1.2.1) for the Lévy area between two discrete stochastic processes.

A.2 Derivation of the Matrix Form of the Dirichlet Energy

In this section, we show that the Dirichlet energy for matrix $X_{N\times p}$ with adjacency matrix $A_{N\times N}$ has the following equivalent forms as defined in (3.1.1):

$$\|X\|^2_{\mathcal{D},G} = \frac{1}{2}\sum_{i,j}\|x^{(i)} - x^{(j)}\|_2^2 A_{i,j} \equiv \mathrm{tr}(X^T L X).$$

we can first rewrite the right-hand side as

$$\begin{aligned}
&\frac{1}{2}\sum_{i,j}\|x^{(i)} - x^{(j)}\|_2^2 A_{i,j} \\
&= \frac{1}{2}\sum_{i=1}^{n}\sum_{j=1}^{n}(x^{(i)} - x^{(j)})^T A_{i,j}(x^{(i)} - x^{(j)}) \\
&= \frac{1}{2}\sum_{i=1}^{n}\sum_{j=1}^{n}({x^{(i)}}^T A_{i,j}x^{(i)} - {x^{(j)}}^T A_{i,j}x^{(i)} - {x^{(i)}}^T A_{i,j}x^{(j)} + {x^{(j)}}^T A_{i,j}x^{(j)})
\end{aligned} \tag{A.2.1}$$

By noticing that $A_{i,j}$ is symmetric, we have

$$\sum_{i=1}^{n}\sum_{j=1}^{n}{x^{(i)}}^T A_{i,j}x^{(i)} = \sum_{i=1}^{n}\sum_{j=1}^{n}{x^{(j)}}^T A_{i,j}x^{(j)}.$$

Therefore we can reduce (A.2.1) further by writing

$$\begin{aligned}
\frac{1}{2}\sum_{i,j}\|x^{(i)} - x^{(j)}\|_2^2 A_{i,j} &= \sum_{i=1}^{n}\sum_{j=1}^{n}{x^{(i)}}^T A_{i,j}x^{(i)} - \sum_{i=1}^{n}\sum_{j=1}^{n}{x^{(i)}}^T A_{i,j}x^{(j)} \\
&= \sum_{i=1}^{n}{x^{(i)}}^T x^{(i)}\sum_{j=1}^{n}A_{i,j} - \mathrm{tr}(X^T A X).
\end{aligned} \tag{A.2.2}$$

Recall the definition of the degree vector $\mathbf{d} = A\mathbf{1}$ where the $i-$th element is essentially $\sum_{j=1}^{n}A_{i,j}$, let D denote the degree matrix which has diagonal being $\mathbf{d}$ and zero elsewhere, we can rewrite (A.2.2) as

$$\begin{aligned}
\frac{1}{2}\sum_{i,j}\|x^{(i)} - x^{(j)}\|_2^2 A_{i,j} &= \mathrm{tr}(X^T D X) - \mathrm{tr}(X^T A X) \\
&= \mathrm{tr}(X^T L X),
\end{aligned}$$

where we use the definition of the Laplacian matrix of $L = D - A$. This is in the same form as we defined in (3.1.1).

Appendix B

Supplementary Data

B.1 Dataset Details

In Table B.1, we summarise the Bloomberg tickers and names of all the futures contracts we used in our portfolio.

Bloomberg Ticker	**Contract Name**	**Market Class**
future_bo1_comdty	CBOT Soybean Oil Future	Ags
future_sm1_comdty	CBOT Soybean Meal Future	Ags
future_sb1_comdty	NYBOT CSC Number 11 World Sugar Future	Ags
future_rr1_comdty	Rough Rice Future	Ags
future_o_1_comdty	Oats Future	Ags
future_mw1_comdty	MGE Red Wheat Future	Ags
future_kw1_comdty	KCBT Hard Red Winter Wheat Future	Ags
future_kc1_comdty	NYBOT CSC C Coffee Future	Ags
future_jo1_comdty	Orange Juice (RTH) Future	Ags
future_w_1_comdty	CBOT Wheat Future	Ags
future_c_1_comdty	CBOT Corn Future	Ags
future_cc1_comdty	NYBOT CSC Cocoa Future	Ags
future_da1_comdty	Class III Milk Future	Ags
future_ct1_comdty	NYBOT CTN Number 2 Cotton Future	Ags
future_cl1_comdty	NYMEX Light Sweet Crude Oil Future	Energy
future_co1_comdty	ICE Brent Crude Oil Future	Energy
future_ng1_comdty	NYMEX Henry Hub Natural Gas Future	Energy
future_cf1_index	Euronext CAC 40 Index Future	Equity
future_nq1_index	CME E-Mini NASDAQ 100 Index Future	Equity
future_vg1_index	Eurex EURO STOXX 50 Future	Equity
future_hi1_index	HKG Hang Seng Index Future	Equity
future_gx1_index	Eurex DAX Index Future	Equity
future_es1_index	CME E-Mini Standard & Poor's 500 Future	Equity
future_pa1_comdty	NYMEX Palladium Future	Metals
future_pl1_comdty	NYMEX Platinum Future	Metals
future_hg1_comdty	COMEX Copper Future	Metals
future_si1_comdty	COMEX Silver Future	Metals
future_gc1_comdty	COMEX Gold 100 Troy Ounces Future	Metals

Table B.1: Futures Contracts from Bloomberg

B.2 Supplementary Skewness Plots Across Time Horizons

In Figure B.1, we include skewness plots for additional network momentum models not presented in Section 4.5.4.

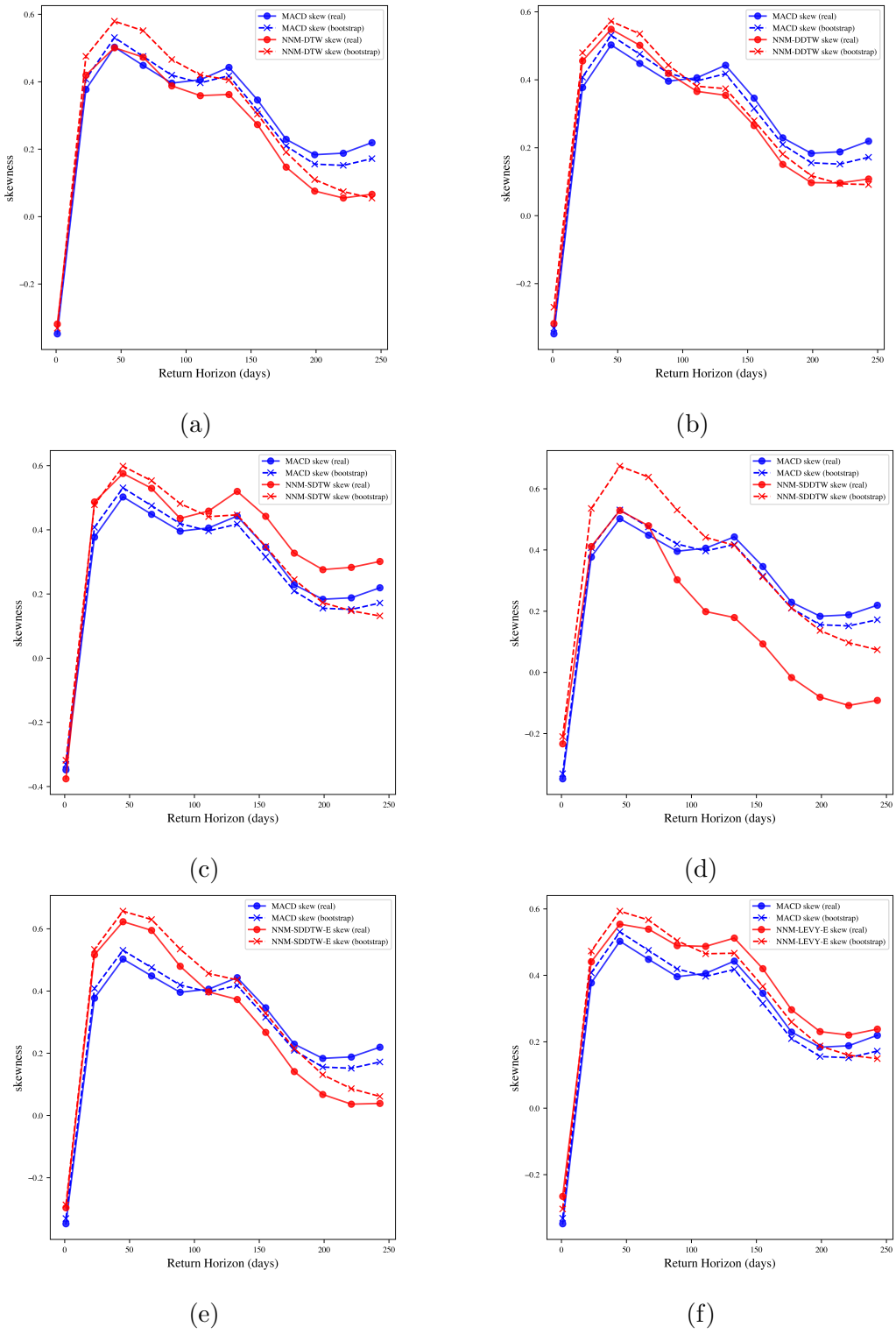


Figure B.1: Supplementary plots of skewness in the returns of the network momentum model over various periods, compared to those of the time series momentum model, using different lead-lag detection models.

Appendix C

Motivation and Methodology for Using Bootstrapping in Backtesting

C.1 Limitations of Historical Backtesting

Backtesting is a pivotal method for evaluating the efficacy of investment strategies by utilising metrics such as profit and loss (PnL), Sharpe ratio, volatility, and maximum drawdown. The most prevalent and intuitive method, often termed 'historical backtesting', involves assessing strategies against actual price movements over a specified historical period. This approach is favoured primarily due to its straightforward implementation and ease of interpretation. Practitioners can directly analyse the performance of strategies and conduct more nuanced analyses, such as identifying industries where strategies performed the best. However, historical backtesting is subject to significant limitations, as discussed in [59, 4, 32]. We summarise these pitfalls as follows:

1. **Limited Scenario Testing:** Historical backtesting often tests only a single price trajectory, giving practitioners limited information to understand the underlying reasons for a strategy's performance. The results are also sensitive to minor changes in the input data. For example, Figure C.1 shows the cumulative return of the NASDAQ 100 mini futures from September 2016 to May 2024. If the strategy did not hold positions during the 10 days with the largest price increases, the final value of these futures would be approximately 34% lower than the market; conversely, avoiding position in the 10 days with the largest price drawdowns would result in an increase of about 40%. This demonstrates how sensitive the performance of our strategy can be to one single path. Although performance improvement can often be achieved by tuning parameters, this can lead to 'overfitting', where adjustments to enhance historical performance do not necessarily translate to future gains.

2. **Past Performance and Future Predictability:** The patterns and trends of price movements observed in the past are not guaranteed to recur in the future. Consequently, past performance may not be indicative of future returns. This is especially problematic for trend-following strategies that may generate significant profits from a few large price movements. If such events are statistically unlikely to occur with similar frequency in the future, practitioners may be misled by historical backtesting results.

One desired property of backtesting should be the availability of multiple paths to evaluate the strategy and check its robustness [32]. This approach enables practitioners to

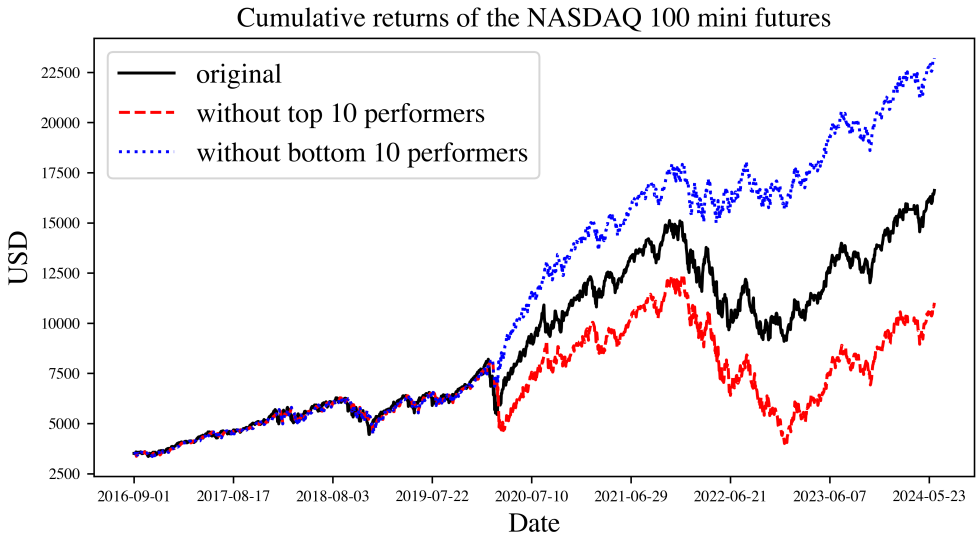


Figure C.1: The sensitivity of historical backtesting results to the price trajectory

end up with a series of performance metrics, rather than a single data point. Practitioners can then apply inferential statistics to analyse these sequences. For example, if comparing the performances of two different models, resampling methods and hypothesis tests can be used to check whether the differences between them are statistically significant and if they come from the same distribution. As discussed in [49], resampling methods are less prone to overfitting. The historical backtesting can serve as an aid, providing a demonstration of the strategy's performance on real price trajectories, but it should be supplemented with these more robust statistical approaches.

C.2 Stationary Bootstrap

C.2.1 Classical Bootstrap and Motivation for Block Bootstrap

Efron has introduced the classical bootstrapping method in [19], a robust nonparametric approach for approximating the sampling distribution of identically and independently distributed observations. As detailed in [51], consider a time series without autocorrelation, $X_1, \cdots, X_N$, and statistic $T(X_1, \cdots, X_N)$. The joint distribution $q(X_1, \cdots, X_N)$ can be expressed through the product of a single probability distribution q_0 as:

$$q(X1, \cdots, X_N) = \prod_{i=1}^{n} q_0(X_i).$$

Under the assumption of independence, q_0 can be approximated by the empirical distribution q_{emp}, defined as:

$$q_{\text{emp}}(X) = \frac{1}{n} \sum_{i=1}^{n} \delta(X - X_i),$$

where $\delta(x)$ is the Dirac delta function, and $N \gg 1$. Sampling from the empirical distribution is equivalent to drawing samples from the series $X_1, \cdots, X_N$ with replacement. In practice, to bootstrap B sets of data, we sample N new data points $X_1^{*(b)}, \cdots, X_N^{*(b)}$ for $b = 1, \cdots, B$. This allows estimation of statistics $T(X_1, \cdots, X_N)$ via $T(X_1^{*(b)}, \cdots, X_N^{*(b)})$.

Nevertheless, this method assumes no autocorrelation in the observed series and is typically applied to univariate time series. Financial time series often exhibit significant

autocorrelation and volatility clustering, with marked inter-correlation among markets. Figure C.2 shows an example of what one should aim for in the bootstrapping method. It displays the price trajectories of two markets that move closely together, highlighting their strong correlation. In such cases, it's important to maintain this close relationship in the bootstrapped data, ensuring that the simulated prices continue to reflect the same level of dependency between these markets. These characteristics motivate the use of another bootstrapping scheme to preserve the autocorrelation of each market and the covariance among different market returns in multivariate series.

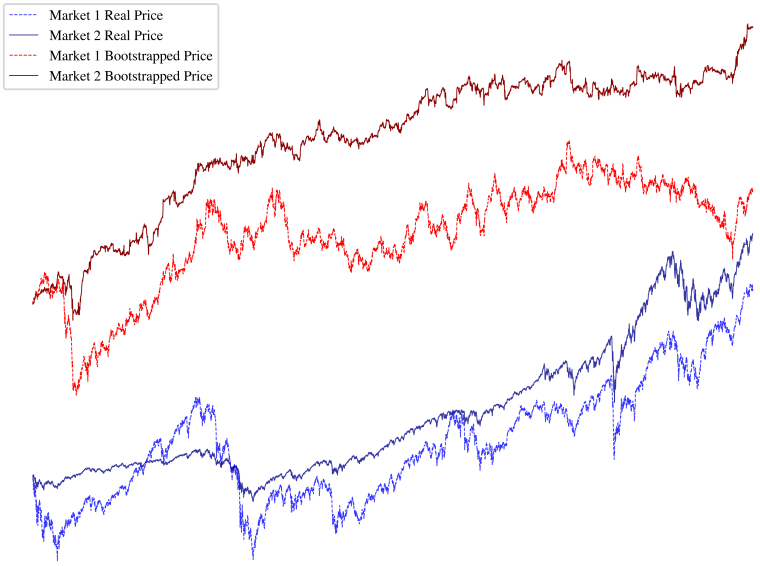


Figure C.2: Desired Multivariate Bootstrap Outcomes Preserving Inter-Market Autocorrelation and Covariance: EURO STOXX 50 Index Futures (Market 1) and CME E-mini S&P 500 Index Futures (Market 2), June 2002 to June 2024.

A popular alternative is the 'blocked bootstrapping' [38, 42, 53]. Intuitively, this method involves splitting the contiguous samples from a multivariate time series into blocks of varying lengths to incorporate the temporal dependence, then sampling these blocks with replacement and reassembling them to construct a pseudo-time series $X_1^*, \cdots, X_N^*$.

C.2.2 Methods of Stationary Bootstrap

We now rigorously present the concept of one variant of blocked bootstrapping, known as the stationary bootstrap, which was proposed in [54]. This approach preserves the stationary property [54, Proposition 1, page 1304] and similar statistical characteristics of the observed time series.

Given an observed series $X_1, \cdots, X_N$ that satisfies the following two assumptions:

1. The series is strictly stationary, meaning that parameters such as mean and variance are time-translation invariant.

2. The series is weakly dependent, implying that the time series is asymptotically uncorrelated as time progresses.

To bootstrap the pseudo-time series $X_1^*, \cdots, X_N^*$, we begin by setting a parameter $p \in (0,1)$. Let $L_1, L_2, \cdots$ be a sequence of independent and identically distributed random variables following a geometric distribution, where $P(L_i = m) = (1-p)^{m-1}p$ for $m = 1, 2, \cdots$. Here, L_i determines the length of the i-th block of consecutive values. Independently of the X_i and L_i, let $I_1, I_2, \cdots$ be a sequence of independent and identically distributed random variables with a discrete uniform distribution over $\{1, \cdots, N\}$, which determines the index of the first bootstrapped sample from the original series in each block. For instance, B_{I_1,L_1} contains the observations $\{X_{I_1}, X_{I_1+1}, \cdots, X_{I_1+L_1-1}\}$, constituting the first L_1 observations in the pseudo-time series. The second block, with L_2 observations, is determined by $X_{I_2}, \cdots, X_{I_2+L_2-1}$. In the case of $j > N$, X_j is defined as X_i where $i = j (\mathrm{mod}\, N)$. This process can be iterated to generate an arbitrary length; however, in our case, we terminate once N observations have been bootstrapped for the pseudo-time series. As discussed in [54], an equivalent and simpler algorithm can be summarised in Table C.1. It has been demonstrated in [54, Theorem 1, page 1305] that as $p \to 0$ and $Np \to \infty$, the bootstrapped estimate of the variance $\hat{\sigma}^2$ converges to the true variance σ^2 in probability.

In this algorithm, one parameter must be manually selected—the geometric distribution parameter p, which represents the average block size. The parameter p significantly influences the estimated value of the confidence interval; thus, selecting an appropriate p is essential to address the correlation effects among the original time series data. In practical applications, previous research on block bootstrap methods indicates that the average block size b should increase at a specific rate relative to the size of the time series N. For instance, [53, Theorem 2, page 5] recommends selecting b such that $bN^{-1/3} \to 0$ as $N \to \infty$ and $b \to \infty$. Furthermore, [55, 2] introduce an automatic block size selection method for stationary bootstrap, aiming to minimise the mean-squared error of the estimated variance from the bootstrapped datasets.

Table C.1 Algorithm for Stationary Bootstrap

Require: Time series X with length N: $X_1, \ldots, X_N$.
Ensure: Bootstrapped series X^* with length N.
1: Calculate the optimal block length B^* from the auto-selection algorithm in [55, 2] and define the parameter $p = \frac{1}{B^*}$
2: Initialise X^* as an empty series.
3: Choose t_1 uniformly at random from $[1, N]$ and set $X_1^* = X_{t_1}$.
4: Set $i = 2$.
5: **while** $i \leq N$ **do**
6: Draw u from $U \sim$ Uniform(0,1).
7: **if** $u < p$ **then**
8: Choose t_i uniformly at random from $[1, N]$.
9: **else**
10: Set $t_i = (t_{i-1} + 1) \bmod N$.
11: **end if**
12: Set $X_i^* = X_{t_i}$.
13: Set $i = i + 1$.
14: **end while**
15: **return** X^*.

Bibliography

[1] U. Ali and D. Hirshleifer, *Shared analyst coverage: Unifying momentum spillover effects*, Journal of Financial Economics, 136 (2020), pp. 649–675.

[2] D. N. P. Andrew Patton and H. White, *Correction to "automatic block-length selection for the dependent bootstrap" by d. politis and h. white*, Econometric Reviews, 28 (2009), pp. 372–375.

[3] S. G. Badrinath, J. R. Kale, and T. H. Noe, *Of shepherds, sheep, and the cross-autocorrelations in equity returns*, The Review of Financial Studies, 8 (1995), pp. 401–430.

[4] D. H. Bailey, J. Borwein, M. López de Prado, and Q. J. Zhu, *Pseudo-mathematics and financial charlatanism: The effects of backtest overfitting on out-of-sample performance*, Notices of the American Mathematical Society, 61 (2014), pp. 458–471.

[5] N. Barberis, A. Shleifer, and R. Vishny, *A model of investor sentiment*, Journal of financial economics, 49 (1998), pp. 307–343.

[6] L. Basnarkov, V. Stojkoski, Z. Utkovski, and L. Kocarev, *Lead–lag relationships in foreign exchange markets*, Physica A: Statistical Mechanics and its Applications, 539 (2020), p. 122986.

[7] J. Baz, N. Granger, C. Harvey, N. Roux, and S. Rattray, *Dissecting investment strategies in the cross section and time series*, SSRN Electronic Journal, (2015).

[8] S. Bennett, M. Cucuringu, and G. Reinert, *Lead-lag detection and network clustering for multivariate time series with an application to the us equity market*, 2022.

[9] V. W. Berger and Y. Zhou, *Kolmogorov–smirnov test: Overview*, Wiley statsref: Statistics reference online, (2014).

[10] M. Billio, M. Getmansky, A. W. Lo, and L. Pelizzon, *Econometric measures of connectedness and systemic risk in the finance and insurance sectors*, Journal of financial economics, 104 (2012), pp. 535–559.

[11] M. J. Brennan, N. Jegadeesh, and B. Swaminathan, *Investment analysis and the adjustment of stock prices to common information*, The Review of Financial Studies, 6 (1993), pp. 799–824.

[12] J. Y. Campbell, A. W. Lo, A. C. MacKinlay, and R. F. Whitelaw, *The econometrics of financial markets*, Macroeconomic Dynamics, 2 (1998), pp. 559–562.

[13] Á. Cartea, M. Cucuringu, and Q. Jin, *Detecting lead-lag relationships in stock returns and portfolio strategies*, Available at SSRN, (2023).

[14] S. Chen, B. Ma, and K. Zhang, *On the similarity metric and the distance metric*, Theoretical Computer Science, 410 (2009), pp. 2365–2376.

[15] I. Chevyrev and A. Kormilitzin, *A primer on the signature method in machine learning*, 2016.

[16] A. Chui, S. Titman, and K. J. Wei, *Momentum, ownership structure, and financial crises: An analysis of asian stock markets*, wp University of Texas at Austin, (2000).

[17] P. Declerck, *Trend-following and spillover effects*, Available at SSRN 3473657, (2019).

[18] S. Diamond and S. Boyd, *Cvxpy: A python-embedded modeling language for convex optimization*, 2016.

[19] B. Efron, *Bootstrap methods: Another look at the jackknife*, The Annals of Statistics, 7 (1979), pp. 1–26.

[20] A. Fernandez-Perez, I. Indriawan, Y. Tse, and Y. Xu, *Cross-asset time-series momentum: Crude oil volatility and global stock markets*, Journal of Banking & Finance, 154 (2023), p. 106704.

[21] P. Fiedor, *Information-theoretic approach to lead-lag effect on financial markets*, The European Physical Journal B, 87 (2014), pp. 1–9.

[22] W. R. Gebhardt, S. Hvidkjaer, and B. Swaminathan, *Stock and bond market interaction: Does momentum spill over?*, Journal of Financial Economics, 75 (2005), pp. 651–690.

[23] K. Grobys, J. Ruotsalainen, and J. Äijö, *Risk-managed industry momentum and momentum crashes*, Quantitative Finance, 18 (2018), pp. 1715–1733.

[24] W. Guthrie, A. Filliben, and T. Tarvainen, *Nist/sematech e-handbook of statistical methods*, Process Modeling, (2013).

[25] L. G. Gyurkó, T. Lyons, M. Kontkowski, and J. Field, *Extracting information from the signature of a financial data stream*, 2014.

[26] D. Haesen, P. Houweling, and J. van Zundert, *Momentum spillover from stocks to corporate bonds*, Journal of Banking & Finance, 79 (2017), pp. 28–41.

[27] H. Hong and J. C. Stein, *A unified theory of underreaction, momentum trading and overreaction in asset markets*, Journal of Finance, LIV (1999), pp. 2143–2184.

[28] K. Hou, *Industry information diffusion and the lead-lag effect in stock returns*, The review of financial studies, 20 (2007), pp. 1113–1138.

[29] B. Hurst, Y. H. Ooi, and L. H. Pedersen, *A century of evidence on trend-following investing*, Available at SSRN 2993026, (2017).

[30] N. Jegadeesh and S. Titman, *Returns to buying winners and selling losers: Implications for stock market efficiency*, The Journal of finance, 48 (1993), pp. 65–91.

[31] ——, *Profitability of momentum strategies: An evaluation of alternative explanations*, The Journal of finance, 56 (2001), pp. 699–720.

[32] J. Joubert, D. Sestovic, I. Barziy, W. Distaso, and M. López de Prado, *The three types of backtests*. Available at SSRN, 7 2024.

[33] V. Kalofolias, *From data to structures: graph learning under smoothness assumptions and applications in data science*, PhD thesis, EPFL, Lausanne, 2016.

[34] V. Kalofolias, *How to learn a graph from smooth signals*, in Artificial intelligence and statistics, PMLR, 2016, pp. 920–929.

[35] M. G. Kendall, *A new measure of rank correlation*, Biometrika, 30 (1938), pp. 81–93.

[36] E. J. Keogh and M. J. Pazzani, *Derivative dynamic time warping*, in Proceedings of the 2001 SIAM International Conference on Data Mining, Chicago, IL, USA, April 5–7 2001, Society for Industrial and Applied Mathematics, pp. 1–11.

[37] H. Kirchhoff and A. Lerch, *Evaluation of features for audio-to-audio alignment*, Journal of New Music Research, 40 (2011), pp. 27–41.

[38] H. R. Künsch, *The jackknife and the bootstrap for general stationary observations*, The Annals of Statistics, 17 (1989), pp. 1217–1241.

[39] A. Levine and L. H. Pedersen, *Which trend is your friend?*, Financial Analysts Journal, 72 (2016), pp. 51–66.

[40] B. Lim, S. Zohren, and S. Roberts, *Enhancing time series momentum strategies using deep neural networks*, 2020.

[41] L. X. Liu and L. Zhang, *Momentum Profits, Factor Pricing, and Macroeconomic Risk*, The Review of Financial Studies, 21 (2008), pp. 2417–2448.

[42] R. Y. Liu, K. Singh, et al., *Moving blocks jackknife and bootstrap capture weak dependence*, Exploring the limits of bootstrap, 225 (1992), p. 248.

[43] A. W. Lo and A. C. MacKinlay, *When are contrarian profits due to stock market overreaction?*, The Review of Financial Studies, 3 (1990), pp. 175–205.

[44] G. Marti, S. Andler, F. Nielsen, and P. Donnat, *Exploring and measuring non-linear correlations: Copulas, lightspeed transportation and clustering*, in NIPS 2016 Time Series Workshop, PMLR, 2017, pp. 59–69.

[45] R. J. Martin, *Design and analysis of momentum trading strategies*, 2023.

[46] G. Mateos, S. Segarra, A. G. Marques, and A. Ribeiro, *Connecting the dots: Identifying network structure via graph signal processing*, IEEE Signal Processing Magazine, 36 (2019), p. 16–43.

[47] W. Meert, K. Hendrickx, T. Van Craenendonck, P. Robberechts, H. Blockeel, and J. Davis, *Dtaidistance*, Oct. 2022.

[48] T. J. Moskowitz and M. Grinblatt, *Do industries explain momentum?*, The Journal of finance, 54 (1999), pp. 1249–1290.

[49] F. Musciotto, L. Marotta, S. Miccichè, and R. Mantegna, *Bootstrap validation of links of a minimum spanning tree*, Physica A: Statistical Mechanics and its Applications, 512 (2018), p. 1032–1043.

[50] M. Müller, *Dynamic time warping*, Information Retrieval for Music and Motion, 2 (2007), pp. 69–84.

[51] Y. Nishikawa, J. Takahashi, and T. Takahashi, *Stationary bootstrap: A refined error estimation for equilibrium time series*, 2021.

[52] D. Poh, B. Lim, S. Zohren, and S. Roberts, *Building cross-sectional systematic strategies by learning to rank*, arXiv preprint arXiv:2012.07149, (2020).

[53] D. N. Politis and J. P. Romano, *A circular block-resampling procedure for stationary data*, Purdue University. Department of Statistics, 1991.

[54] D. N. Politis and J. P. Romano, *The stationary bootstrap*, Journal of the American Statistical Association, 89 (1994), pp. 1303–1313.

[55] D. N. Politis and H. White, *Automatic block-length selection for the dependent bootstrap*, Econometric Reviews, 23 (2004), pp. 53–70.

[56] X. Pu, S. Roberts, X. Dong, and S. Zohren, *Network momentum across asset classes*, 2023.

[57] T. Rakthanmanon, B. Campana, A. Mueen, G. Batista, B. Westover, Q. Zhu, J. Zakaria, and E. Keogh, *Addressing big data time series: Mining trillions of time series subsequences under dynamic time warping*, ACM Transactions on Knowledge Discovery from Data (TKDD), 7 (2013), pp. 1–31.

[58] K. G. Rouwenhorst, *International momentum strategies*, The journal of finance, 53 (1998), pp. 267–284.

[59] E. Schumann, *Backtesting*. Available at SSRN, 12 2018. Forthcoming in "Numerical Methods and Optimization in Finance (2nd ed)," by M. Gilli, D. Maringer and E. Schumann.

[60] M. Shokoohi-Yekta, B. Hu, H. Jin, J. Wang, and E. Keogh, *Generalizing dtw to the multi-dimensional case requires an adaptive approach*, Data mining and knowledge discovery, 31 (2017), pp. 1–31.

[61] J. Stübinger and D. Walter, *Using multi-dimensional dynamic time warping to identify time-varying lead-lag relationships*, Sensors, 22 (2022).

[62] G. J. Székely, M. L. Rizzo, and N. K. Bakirov, *Measuring and testing dependence by correlation of distances*, (2007).

[63] W. L. Tan, S. Roberts, and S. Zohren, *Spatio-temporal momentum: Jointly learning time-series and cross-sectional strategies*, arXiv preprint arXiv:2302.10175, (2023).

[64] G. A. Ten Holt, M. J. Reinders, and E. A. Hendriks, *Multi-dimensional dynamic time warping for gesture recognition*, in Thirteenth annual conference of the Advanced School for Computing and Imaging, vol. 300, 2007, p. 1.

[65] A. Varfis, L. Corleto, J. Auger, D. Perrotta, and M. Alvarez, *Lead-lag estimation by means of the dynamic time warping technique*, Research in Official Statistics (European Communities), (2001), p. 5.

[66] D. Vayanos and P. Woolley, *An institutional theory of momentum and reversal*, The Review of Financial Studies, 26 (2013), pp. 1087–1145.

[67] D. Wang, J. Tu, X. Chang, and S. Li, *The lead–lag relationship between the spot and futures markets in china*, Quantitative Finance, 17 (2017), pp. 1447–1456.

[68] F. Wilcoxon, *Individual comparisons by ranking methods*, in Breakthroughs in statistics: Methodology and distribution, Springer, 1992, pp. 196–202.

[69] D. Wu, Y. Ke, J. X. Yu, P. S. Yu, and L. Chen, *Detecting leaders from correlated time series*, in Database Systems for Advanced Applications: 15th International Conference, DASFAA 2010, Tsukuba, Japan, April 1-4, 2010, Proceedings, Part I 15, Springer, 2010, pp. 352–367.

[70] R. Yamamoto, N. Kawadai, and H. Miyahara, *Momentum information propagation through global supply chain networks*, Journal of Portfolio Management, 47 (2021), pp. 197–211.

[71] E. A.-T. Yamani and M. Abuelfadl, *Currency news and international bond markets*, North American Journal of Economics and Finance, (2021).

[72] Y. Zhang, M. Cucuringu, A. Y. Shestopaloff, and S. Zohren, *Dynamic time warping for lead-lag relationships in lagged multi-factor models*, 2023.

[73] J. Zhao and L. Itti, *shapedtw: Shape dynamic time warping*, Pattern Recognition, 74 (2018), pp. 171–184.