A Network Model of Financial Markets

A THESIS PRESENTED
BY
OLIVER RICE

IN PARTIAL FULFILLMENT OF THE REQUIREMENTS
FOR THE DEGREE OF
DOCTOR OF PHILOSOPHY
IN THE SUBJECT OF
FINANCIAL COMPUTING

UNIVERSITY COLLEGE LONDON (UCL)
LONDON, UNITED KINGDOM
JANUARY 2015

I, Oliver Rice confirm that the work presented in this thesis is my own. Where information has been derived from other sources, I confirm that this has been indicated in the thesis.

Advisor: Dr. Robert Smith Oliver Rice

A Network Model of Financial Markets

Abstract

This thesis introduces a network representation of equity markets. The model is based on the premise that assets share dependencies on abstract 'factors' resulting in exploitable patterns among asset price levels. The network model is a collection of long-run market trends estimated by a 3 layer machine learning framework. The network model's comprehensive validity is established with 2 simulations in the fields of algorithmic trading, and systemic risk.

The algorithmic trading validation applies expectations derived from the network model to estimating expected future returns. It further utilizes the network's expectations to actively manage a theoretically market neutral portfolio. The validation demonstrates that the network model's portfolio generates excess returns relative to 2 benchmarks. Over the time period of April, 2007 to January, 2014 the network model's portfolio for assets drawn from the S&P/ASX 100 produced a Sharpe ratio of 0.674. This approximately doubles the nearest benchmark.

The systemic risk validation utilized the network model to simulate shocks to select market sectors and evaluate the resulting financial contagion. The validation successfully differentiated sectors by systemic connectivity levels and suggested some interesting market features. Most notable was the identification of the 'Financials' sector as most systemically influential and 'Basic Materials' as the most systemically dependent. Additionally, there was evidence that 'Financials' may function as a hub of systemic risk which exacerbates losses from multiple market sectors.

Contents

Listing of figures

It is a very sad thing that nowadays there is so little useless information.

- Oscar Wilde

1 Introduction

In this chapter the thesis statement is presented. The motivations for testing the thesis and contributions of the work are then explained. The introduction ends with an outline of subsequent chapters' contents.

1.1 Thesis Statement

> We assert that assets in financial markets are mutually guided by arbitrarily defined external 'factors'. Furthermore, shared dependencies create emergent relationships among assets' price levels. We aim to leverage the theorized inter-asset associations into a meaningful network representation of equities markets which enables unique and valuable analyses.

1.2 Motivation

This thesis introduces an inter-disciplinary framework for analysing relationships among equities. The primary motivation for the model is to identify aspects of market dynamics which are less apparent using traditional techniques.

The associative nature of the proposed network model does not attempt to directly model price movements. Instead, it models relationships among price levels in a large portion of a market's assets. The outcome is a general model capable of accurately representing facets of market behaviour. For example, the network model is able to group assets by response to information shocks, estimate market exposure, determine current market efficiency, and predict price movements for every asset it contains.

A primary advantage of representing markets as networks is the implied consideration of interactions among assets. Representing markets as networks provides an intuitive approach to modelling elements of market dynamics such as systemic risk. It further creates an additional avenue for predicting asset price movements and mitigating market risk in portfolios.

Statistical Arbitrage[9] or StatArb is an algorithmic trading strategy which models assets along a similar premise. StatArb assesses linear relationships among a target asset and several hand-selected assets, typically within in a shared market sector. It's statistical means of identifying significant relationships eliminates the possibility of increasing the model's size beyond a small, isolated, group of similar assets. The limitation inhibits incorporation of all potentially relevant assets in the model. The proposed network model avoids this problem by making all assets available to the model and porting significance identification to machine learning methods.

In summary, the primary motivation for creating a network model of equity markets is to provide a flexible framework for analysing markets in ways which may not be possible using existing approaches. The modelling technique opens up numerous avenues for improving our understanding of markets.

1.3 Contribution

The primary contribution of this thesis is a framework for generating associative models of equity markets which include all assets in a market. The work suggests that modelling assets as a network of inter-relationships can aid in predicting global market reactions to partially realized information.

Secondary contributions include several applications of the model. There are multiple applications explored in later chapters. One covered technique uses the model to identify similarity among assets. Since the proposed framework values assets in terms of each other, analysing the relationships leads to a highly dimensional vector space representation. Comparing relative locations of assets in this space indicates a similarity which loosely categorizes assets by market sector.

The automatic clustering of assets by market sector is a two way validation. It validates the model's output by grouping assets according to a well known construct and re-establishes market sectors as the classification method most likely to indicate similarity in assets' pricing.

A more comprehensive application has direct implications for trading assets. These contributions are the generation of future expected price movements and evaluation of a portfolio's long-run market exposure. Together, these applications are demonstrated out-of-sample on 2 stock exchanges. This provides a simultaneous validation on multiple aspects of the network model's expectations. The implemented strategy is intended as a verification that network representation of markets yields out-of-sample predictability as opposed to a practical trading strategy. Notwithstanding, elements of the strategy hint at a potential for commercialization.

The final application of the network model was to assess systemic risk within a selected market. The network was utilized to simulate shocks to multiple market sectors and their expected perpetuation through the market. The method uncovers interesting relationships among market sectors and clearly identifies the 'Financials' sector as exceedingly interconnected. Moreover, this was discovered from growth period training data implying that the network model could aid in identifying systemic dependencies ex-ante.

In summary, this thesis contributes techniques to:

1. Model financial markets as associative networks
2. Forecast movements in asset price levels
3. Create theoretically market neutral equity portfolios
4. Model systemic risk in financial markets

1.4 Discussion

Recent trends in financial computing show machine learning and artificial intelligence expanding the domain of what is possible. A key benefit machine learning over statistical methods is that all data can be made available to the model. The programmatic construct then decides what is, or isn't, relevant. One trade-off is that the complexity inherent in many machine learning techniques potentially gives them black box[1] status.

Obfuscating the operations performed by the process is dangerous because the system's limitations may not be known by the end user. One could make the argument that statistical methods are equally prone to misuse by inexperienced users. Despite this, the notably larger user-base of statistical modelling techniques and abundance of available literature makes the concern practically accurate if not technically so. In summary, both statistical and machine learning methods are embodied by unique advantages and limitations. While these avenues overlap in some domains, distributing the workload between the regimes to maximize complimentary benefit was a key consideration throughout this research.

Ensemble learning is a subset of machine learning where multiple models factor into a larger decision making process. In practice, a large selection of diverse models yields the best results[51]. The proposed modelling framework for financial markets applies this concept. It does so by creating thousands of weak learners seeking structure among randomly selected assets. These sub-networks are highly diverse in that the randomly selected assets represent unique pieces of the market. These sub-networks are stitched together into a set of models which aim to accurately reflect long-run relationships in equity markets.

The concept of using a network model to identify associations among assets is not a controversial one. The expected point of contention for most readers is the ability to create a model exhibiting this behaviour when implemented on an entire market. The primary difficulty in developing a global network model is maintaining significance when the number of assets is large. For this reason we developed a new approach to establishing relationship significance through iterative sampling, signal processing, and machine learning. To reduce the potential for critical failure points the approach is highly distributed.

[1] A process in which inputs and outputs are observable while inner workings are obscured

1.5 Structure

Chapter 2 Literature Review & Background

The chapter establishes a background in the fields most relevant to the project. The background information is distinguished from the literature review as it does not necessarily represent state-of-the-art research. The literature review covers current works focusing on algorithmic trading, artificial markets, and multi-objective genetic algorithms. Reviews of modern literature for each field are interspersed throughout the background text to maintain conceptual continuity.

Chapter 3 The Network Model

The process of creating a network model of a market is shown from start to finish. The 3 layer structure of the model is introduced and demonstrated using both pseudo and functional code. An explanation of what each layer achieves and an overview of its intent is presented.

Chapter 4 Sensor Creation: Layer 1

The sensor creation layer, or layer 1, creates thousands of weak learners, i.e. sensors, of the market. Each sensor is expected to contain a portion of information relating to markets that is shared by other sensors and some unique information. The process is tested in an artificial market.

Chapter 5 Sensor Evaluation: Layer 2

Evaluating the sensors for quality yields a metric for identifying shared and unique relationships uncovered by the sensors. The outputs are passed forward to the sensor processing layer.

Chapter 6 Sensor Processing: Layer 3

The sensor processing layer considers the trade-off among all evaluation criteria and sensors. It then computes a smaller number of sensor 'portfolios' which combine sensors to balance the criteria. The results are aggregated to generate the network model. The process is validated in both the artificial market and an application in determining similarity among assets in the New York Stock Exchange.

Chapter 7 Comprehensive Validations

There are 2 comprehensive validations of the network model. The first, creates an algorithmic trading strategy from the network's expectations. The second measures the level of systemic risk among market sectors that is implied by the network. Both tests support the network model's validity.

Chapter 8 Conclusions & Future Research

Conclusions of the research are presented along with areas for further development. Some conclusions relate directly to the network model while others focus on the applications used as validation methods throughout the work. Future research including possible areas of study drawn from theoretical underpinnings and practical applications of the model are also outlined.

The techniques I developed for studying turbulence, like weather, also apply to the stock market.

- Benoit Mandelbrot

2

Literature Review & Background

The literature review and background chapter is intended to provide a reference to round out readers' knowledge ranging from basic concepts to currently emerging research relevant to the thesis. The text is written to be as accessible as possible to all quantitatively minded individuals regardless of current expertise. The areas selected for review are listed below.

1. Time Series Analysis
2. Algorithmic Trading
3. Artificial Markets
4. Modern Portfolio Theory
5. Genetic Algorithms

The time series analysis section begins with basic regression, covers topics in residual diagnostics, and ends with models for stationary and nonstationary time

series. Trends in algorithmic trading are then discussed. The section first outlines current research along with pros and cons of automated trading. Subsequently, true financial arbitrage is explained leading into statistical arbitrage. These concepts are reviewed in depth as they bare the most similarity to the network model's algorithmic trading validation test introduced in later chapters. The remaining sections labelled modern portfolio theory (MPT) and genetic algorithms (GAs) are motivated by the third layer of the network model and the algorithmic trading validation respectively. Contrary to likely expectations, modern portfolio theory is utilized for statistical signal processing in network model refinement while a genetic algorithm is applied to the portfolio realignment problem. Every area of the background and literature review maps to an area involved in the network model's construction or subsequent applications reviewed throughout the thesis.

2.1 Time Series Analysis

2.1.1 Regression

Time series analysis is a set of tools which can be used to produce models of an underlying target, or set of target, time series. Single series regression is the standard starting point. Regression minimizes an error vector between the target, or dependent variable, time series, and regressor, or independent variable, time series [1].

$$y = mx + b + \epsilon \tag{2.1}$$

where

$y =$ dependent variable
$x =$ explanatory variable
$m =$ slope
$b =$ constant
$\epsilon =$ error vector

As the best model of y based on x is determined by minimization of the error vector a method for comparing error among models is needed. Ordinary least squares or

OLS is the stock choice for minimization in many simple regression models.

$$OLS(\epsilon) = \sum_{i=1}^{n} \epsilon_i^2 \tag{2.2}$$

$\epsilon_i = i^{th}$ element in ϵ
$n =$ number of elements in series

Squaring each element of the error vector prior to summation increases the impact of large error terms in parametrization of m and b. Thus, small error spread evenly across all estimations of y is preferred to a few large deviations.

The natural extension of singular regression is multiple regression where multiple dependent variables or regressors are present[64].

$$y = X\beta + \epsilon \tag{2.3}$$

where

$$X = \begin{pmatrix} x_{1,1} & \dots & x_{1,p} \\ \vdots & \ddots & \vdots \\ x_{n,1} & \dots & x_{n,p} \end{pmatrix}$$

$p =$ number of independent variables

$$\beta = \begin{pmatrix} m_1 \\ \vdots \\ m_n \end{pmatrix}$$

The column vector of regression coefficients β is the desired output through minimization of the error term. The most notable difference between the above matrix notation and algebraic form is the absence of a constant. Constant values can be inserted into the model by appending a column to the X matrix containing the value of 1 for all n observations. The regressed coefficient within β when multiplied by the column of 1s returns a constant value for each estimation of y values.

2.1.2 Residual Diagnostics

Consider the following scatter plots of time series with target vectors Y and W against regressors X and Z respectively in the form of equation 2.1.

Given time series $[X, Y]$ and $[Z, W]$, minimizing the ordinary least squares

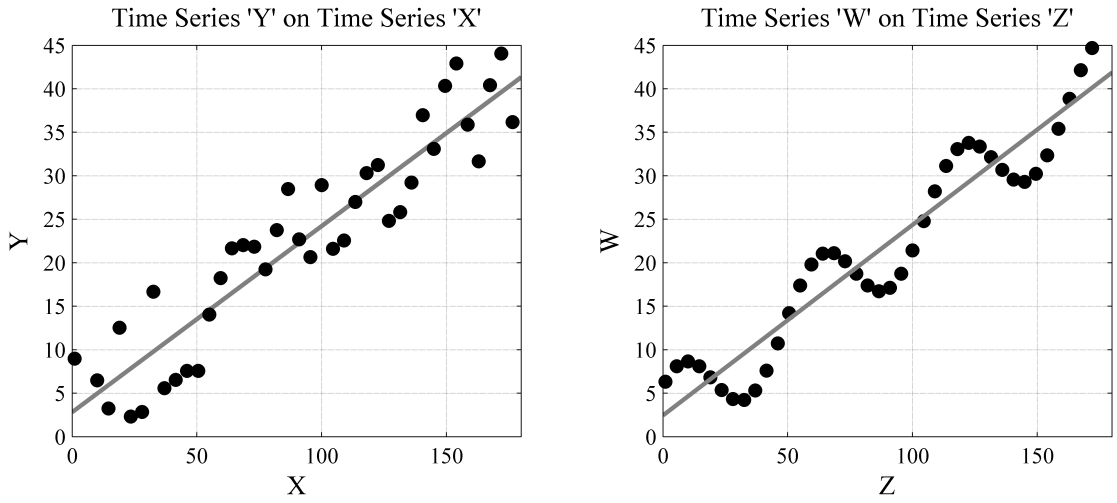


Figure 2.1.1: Background: Linear Regression Models

criterion on the error term of the standard regression model, i.e. least squares regression, yields the grey 'linear prediction' lines in figure 2.1.1. In these cases, equations for the least squares line are identical.

$$Y = 0.219X + 2.45 \tag{2.4}$$

and

$$W = 0.219Z + 2.45 \tag{2.5}$$

To assess the 'goodness-of-fit' the R^2 of each model can be computed. R^2 is related to the ratio of variance in the data explained by the model over the total variance in the data. As such, a perfectly predictive model has an $R^2 = 1$ and a model with no predictive power has $R^2 = 0$. Formally:

$$R^2 = 1 - \frac{\sum_{i=1}^{n} \left(y_i - f(x_i)\right)^2}{\sum_{i=1}^{n} \left(y_i - \sum_{i=1}^{n} \left(\frac{1}{n} y_i\right)\right)^2} \tag{2.6}$$

where

$f(x_i) =$ predicted value of y_i when x_i is applied to model $f()$

The model's residuals are identical in terms of R^2 with values equal to 0.8901. In other words, both models explain the same proportion of summed square residuals

over total residual sum of squares. Despite this, it is clear visually in Figure 2.1.2 that residuals scatter plot of $[X, Y]$ are randomly distributed around the least squares line. In contrast, $[Z, W]$ exhibits structure.

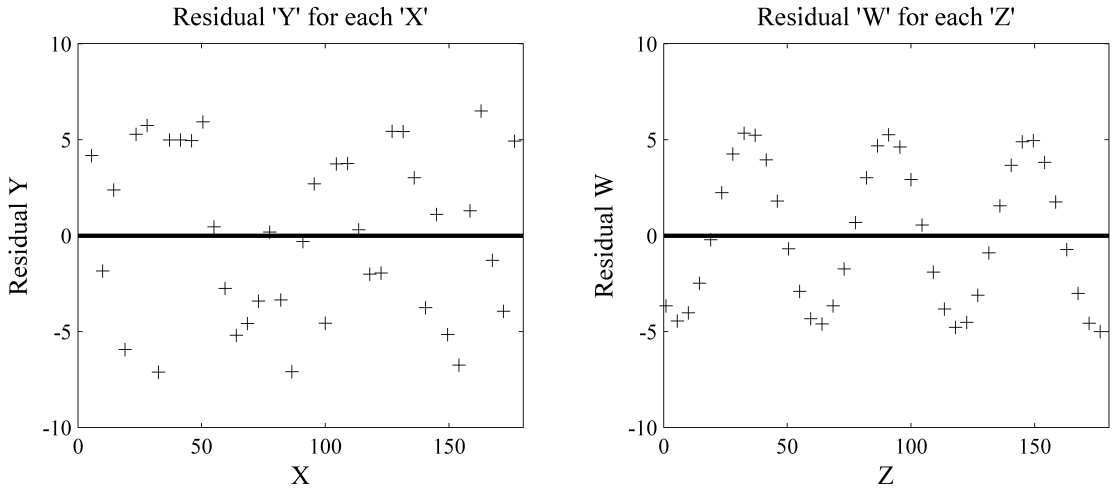


Figure 2.1.2: Background: Serially Autocorrelated and Non-Autocorrelated

For our purposes, residuals of a well specified regression model should match a normal distribution, have constant variance, and exhibit no serial autocorrelation. Normality testing can be performed to a level of confidence contingent on the number of observations using the Jarque-Bera (JB) test [45].

$$JB\left(\epsilon\right) = \frac{n}{6}\left(S^2 + \frac{\left(K-3\right)^2}{4}\right) \tag{2.7}$$

The skewness (S) of residuals can be found via:

$$S\left(\epsilon\right) = \frac{\frac{1}{n}\sum_{i=1}^{n}\left(\epsilon_i - \bar{\epsilon}\right)^3}{\left(\frac{1}{n-1}\sum_{i=1}^{n}\left(\epsilon_i - \bar{\epsilon}\right)^2\right)^{\frac{3}{2}}} \tag{2.8}$$

And kurtosis (K):

$$K\left(\epsilon\right) = \frac{\frac{1}{n}\sum_{i=1}^{n}\left(\epsilon_i - \bar{\epsilon}\right)^4}{\left(\frac{1}{n}\sum_{i=1}^{n}\left(\epsilon_i - \bar{\epsilon}\right)^2\right)^2} \tag{2.9}$$

where

$\bar{\epsilon}$ = mean of ϵ or $\sum_{q=1}^{n}\left(\frac{1}{n}\epsilon_q\right)$

Constant variance can be determined through multiple samplings of continuous residual sections.

$$\sigma^2 = \frac{1}{n}\sum_{i=1}^{n}\left(\epsilon_i - \bar{\epsilon}\right)^2 \tag{2.10}$$

In testing for serial autocorrelation the Durbin-Watson (DW) statistic is applied [42, 43]. Serial autocorrelation is the correlation between a model's residuals at time i, or ϵ_i, and the model's residuals with a 1 period lag i.e. ϵ_{i-1}. Positive autocorrelation suggests positive residuals tend to be followed by additional positive residuals and negatives by negatives. Negative autocorrelation suggests positive residuals are followed by negative residuals and vice versa. Given that a well specified model has randomly distributed residuals, serial autocorrelation of either type is undesirable.

$$DW\left(\epsilon\right) = \frac{\sum_{i=2}^{n}\left(\epsilon_i - \epsilon_{i-1}\right)^2}{\sum_{i=1}^{n}\epsilon_i^2} \tag{2.11}$$

A Durbin-Watson statistic of 2 implies no serial correlation. A statistic significantly less than 2 indicates positive serial autocorrelation, and one significantly higher than 2 suggests negative serial autocorrelation. The maximum range of possible DW statistics is 0 to 4. For context, the example residual series in figure 2.1.2 have statistics of 1.873 and 0.230 for [X,Y] and [W,Z] respectively. This result indicates little serial autocorrelation in [X,Y] and significant positive serial autocorrelation in [W,Z]. The identified structure in the [W,Z] model is a source of concern as structured residuals may result from a poorly specified model.

2.1.3 Cointegration

Consider the relationship between time series in figure 2.1.3.

There does appear to be a mild association between Series 1 and Series 2. To review this relationship, least squares regression was applied yielding equation 2.12.

$$y = 1.334x + 107.5 \tag{2.12}$$

With an $R^2 = 0.80$, the model passes the first diagnostic test. Now note that Series 1 is the estimated number credit card transactions sampled on a daily basis in 1997, while Series 2 represents the spot price for a barrel of crude oil on a minutely basis in

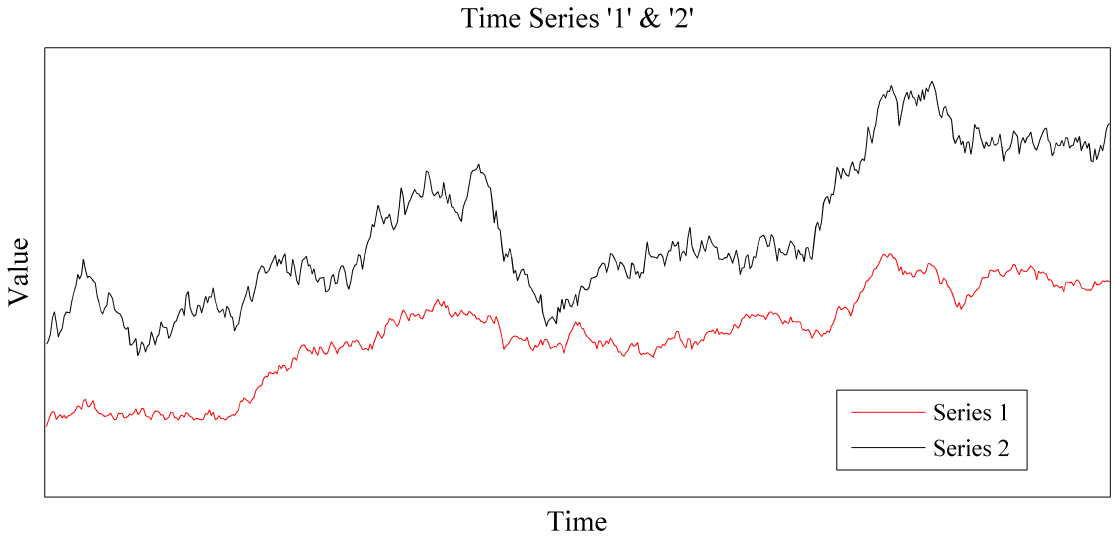


Figure 2.1.3: Background: Cointegration Example

2010. Plainly, the relationship is coincidental. Attempting to specify relationships among unrelated variables is known as spurious regression. To briefly explain the behaviour, note the time series categorization in figure 2.1.4 [12].

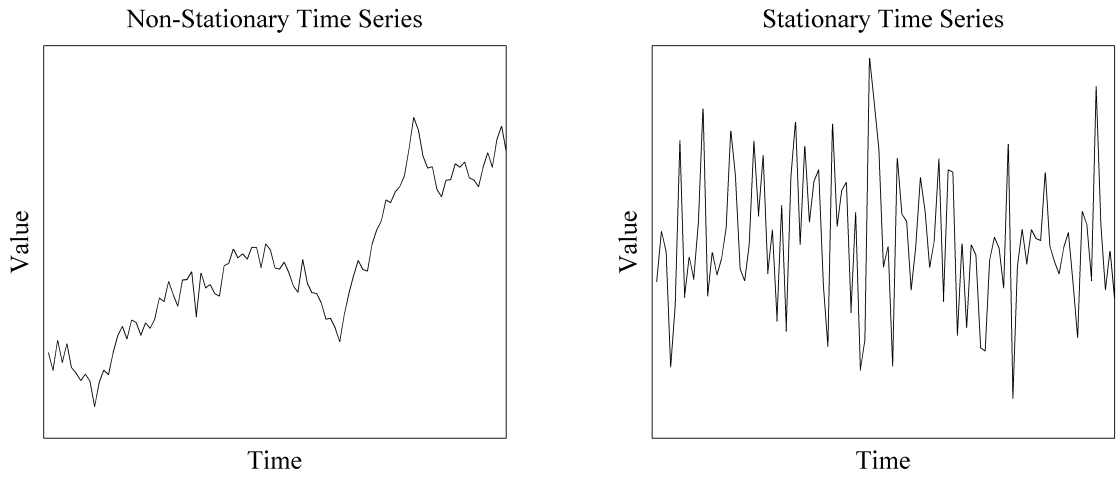


Figure 2.1.4: Background: Introduction to Stationarity

A strictly stationary time series is defined as a stochastic, or random, process with a constant joint probability distribution through time. Weak-sense, or weakly, stationarity relaxes this definition to and encompass a greater set of mean-reverting series. This is characterized in part by a constant mean and variance. In this case,

there is no guarantee that current values are entirely independent from previous values. Weakly stationary series may be autocorrelated with any number of lags. This may direct their tendency to return to a constant or time trend. In contrast, a nonstationary series, or random walk, does not hold a constant mean through time.

In the context of a first order autoregressive model where the only regressor on target series y is a copy of itself offset by lag of 1 i.e. $AR(1)$, the solved parameter β dictates the series' stationarity.

$$y_i = \alpha + \beta y_{i-1} + \varepsilon_i \tag{2.13}$$

Table 2.1.1 Stationarity by β

$\lvert\beta\rvert < 1$	$\lvert\beta\rvert = 1$	$\lvert\beta\rvert > 1$
Weakly Stationary	Non-Stationary	Divergent

Going forward, both nonstationary and divergent time series are referred to as nonstationary. Since weakly stationary time series may be autocorrelated at any lag depth, the Durbin-Watson statistic must be generalized to catch these instances. The Augmented Dickey-Fuller (ADF) test does exactly that by modelling series in the form of equation 2.14.

$$y_i = c + \delta i + \gamma y_{i-1} + \beta_1 \Delta y_{i-1} + ... + \beta_p \Delta y_{i-p} + \epsilon_i \tag{2.14}$$

where

$\Delta y_i = y_i - y_{i-1}$

$\delta =$ time trend coefficient

$p =$ maximum number of lags tested for autocorelation

Including relevant lags p on their corresponding differenced term of y prevents the resulting value from impacting parametrization of γ on level of y_{i-1}. Thus, if the series wanders stochastically, i.e. is nonstationary, Δy_i should be independent from y_{i-1} implying $\gamma = 0$. A weakly stationary time series must be dependent on level of y_{i-1} in order to hold a constant mean requiring $\gamma < 0$.

The null and alternate hypotheses in the ADF test are $\gamma = 1$ and $\gamma < 1$ indicating nonstationary and weakly stationary series respectively.

Returning to the spurious regression example problem; both series appear to be nonstationary. This result is confirmed with and ADF test. The OLS regression model appeared to fit well because the two series happen to share a stochastic drift. While the model seems accurate in the aggregate, on a period to period basis it is at best useless and at worst deceptive. For this reason, only series without stochastic drift are modelled using least squares regression. This then leaves the question of how to model nonstationary time series.

One approach to modelling nonstationary time series is to coerce weak stationarity via differencing.

$$\Delta \mathrm{y}_t = y_t - y_{t-1} \tag{2.15}$$

If the resulting Δy is weakly stationary then the original y series is said to be first order stationary or $I(1)$.

Another approach to modelling relationships among nonstationary time series is via cointegration[25]. A vector autoregressive model (VAR) is a technique for modelling the spread among time series sets. This tools is valuable in determining if a set of time series are cointegrated. For two series x and y to be cointegrated, the following conditions must be met.

$$c = \frac{1}{n} \sum_{i=1}^{n} (x_i + \tau y_i) \tag{2.16}$$

$$c_i = x_i + \tau y_i \tag{2.17}$$

$$k = x + \tau y - c + \epsilon \tag{2.18}$$

where

$\tau =$ coefficient on series Y

The time series k must be weakly stationary with a mean of 0. In other words, some linear combination of the set of time series is stationary around a constant mean c. Take, for instance, the pair of nonstationary time series in figure 2.1.5

The series appear to share a common stochastic drift. Modelling this relationship in

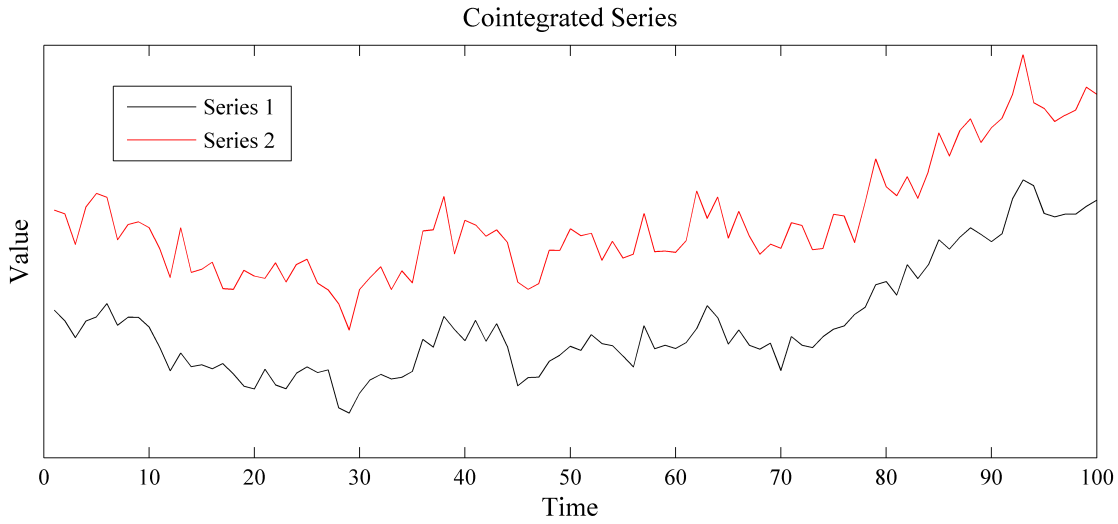


Figure 2.1.5: Background: Introduction to Cointegration

the form of equation 2.18 returns:

$$k = x - \ 0.9367y + 7.0093 + \epsilon_i \tag{2.19}$$

To ensure a well specified model, in addition to the residual tests for least squares regression, the cointegration residuals should also be a weakly stationary time series.

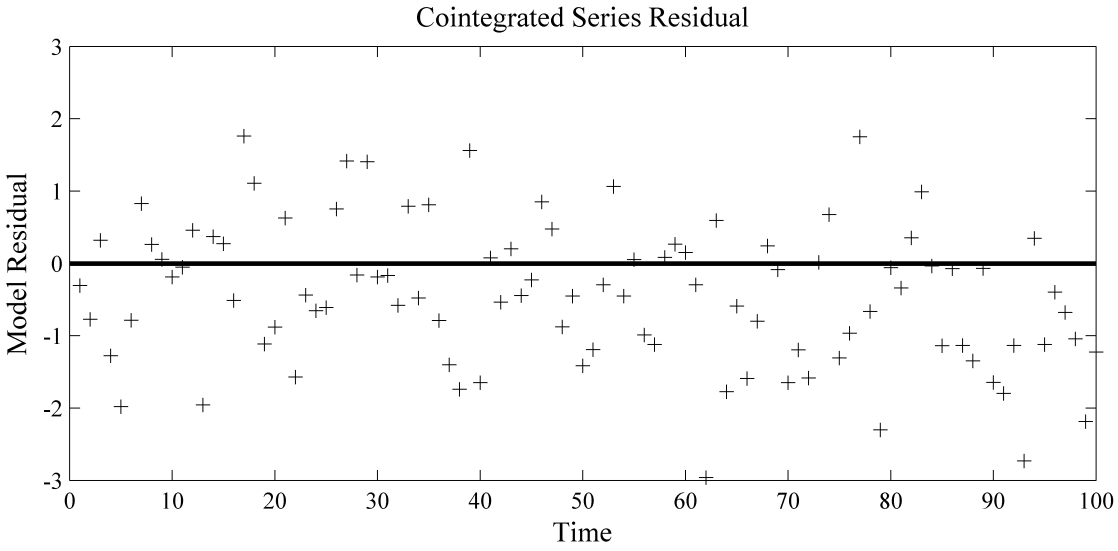


Figure 2.1.6: Background: Cointegration Residuals

Given the high $R^2 = 0.9502$ and residuals passing the aforementioned battery of

tests, the cointegrating relation is well specified. The ADF test is not necessary for cointegration models as their is no requirement for serial autocorrelation.

Table 2.1.2 Cointegration residual tests

Statistic	Purpose	Value	Result
Durbin-Watson	Serial Autocorrelation	2.055	N/A
Jarque-Bera	Normality	p>0.05	Pass
Augmented Dickey-Fuller	Stationarity	p<0.05	Pass

Expectations derived from cointegration are the foundation for a simple algorithmic trading strategy known as pairs trading, a type of statistical arbitrage strategy covered in the next section.

2.2 Algorithmic Trading

As computational resources have become more affordable the prospect of algorithmic or computer automated trading has grown increasingly attractive. The main benefits of algorithmic trading (AT) are founded in speed and consistency. For purely mathematical trading decisions, computers are able to reliably compute trading signals at speeds much faster than their human counterparts. This allows trading entities to enter beneficial positions more quickly while eliminating human error and employee overhead. Moreover, the nature of the computerized system enables trading scalability for little additional cost.

The most automatable trading tools were among the first to be converted into algorithmic trading strategies. Many common tools in technical analysis meet this automation criteria of computationally simple, easily parametrized, highly repetitive tasks which can be described programmatically. Despite their early adoption there is little evidence to indicate any potential for out-of-sample profitability in technical analysis strategies. Fang, Jacobsen, and Qin drew this conclusion after reviewing profitability of 26 technical analysis strategies drawn from earlier work [11] and extending the sampled time period through 2011 [27]. We include this work not only to give instances of early algorithmic trading but also as evidence that AT is merely an implementation of an existing algorithm. If a strategy is invalid when managed by a trader no amount of automation can make it valuable. The exception to this rule is when the profitability of a trading strategy is bound by execution speed.

Although there are many nonsensical uses, AT can provide great value with the correct trading logic. This is particularly true in cases where strategies are computationally expensive or constrained by execution speed. High-frequency trading (HFT), or using computers to make decisions and execute transactions very quickly, is a growing field within algorithmic trading. HFT allows traders to take advantage of extremely time sensitive information which would otherwise expire before a trader could manually take the indicated positions. Given that the proportion of market and limit orders allocatable to algorithmic traders is growing, there is evidence that these agents play a significant role in determining market efficiency [16]. Algorithmic traders are becoming more common in most financial markets including foreign exchange [50], multiple spot markets [34], futures [74], and bonds [18].

Not all algorithmic trading systems are directly intended to produce excess returns. Some, which straddle the boundary, are intended to function as market makers [31] and are actively compensated by exchanges aiming to improve liquidity for their market participants. Others applications of algorithmic trading focus on optimal order execution. An example in equity markets is entering positions while achieving a target price metric such as volume-weighted average price (VWAP) [37]. Algorithmic methods can similarly be useful in execution of large block trades where costly slippage can be minimized by issuing multiple smaller orders in accordance with constrained stochastic optimal control methods [68].

Recent trends in algorithmic trading apply innovative methods or new data sources to derive benefit such as artificial intelligence or text analysis. Artificial intelligence in algorithmic trading is receiving much attention as it is proving both academically and practically tractable. Areas of use include predicting financial asset price movements in both spot [69] and foreign exchange markets [26], predicting macroeconomic impact of events [70], and portfolio optimization [79]. Text analysis is another growing field in algorithmic trading for both increasing returns [30] and reducing risk [29].

The implications of fast decision making and similarities in algorithmic trading strategies have also spurred some unease. Herding is a concern since many strategies tend to mirror recently profitable positions. This behavior can exacerbate otherwise insignificant market movements and cause volatility clustering as the strategy's decisions change [58]. Volatility clustering is highly undesirable from a market participant's perspective and remains a source of contention for regulators.

The network model of equities markets we introduce is partially validated through use of an algorithmic trading strategy. Due to nature of financial markets it is not possible to 'confirm' the correctness of the underlying assumptions inherent in the model. Instead, we support claimed sources of the model's functionality by exploiting it to predict future market movements. By then simulating a trader taking market positions to maximize profitability of these movements we were able to validate the model's accuracy through the profitability of the trades. In other words, by creating an algorithmic trading system based on the proposed network model we determine the model's validity. While other validations are presented, emphasis is put on this trading model throughout the remainder of the thesis as it is the most quantitatively conclusive of the validation methods. Of known existing algorithmic trading strategies, statistical arbitrage shares the most similarity with the validation system. The remainder of the algorithmic trading section outlines the principals guiding statistical arbitrage to be drawn on in later chapters.

2.2.1 Arbitrage

Arbitrage in financial markets exploits price differences of identical instruments among markets to profit from their spread. For example, 2 lemonade stands are on opposite ends of a street. If vendor 1 is willing to buy or sell lemonade for 95 cents per glass, and vendor 2 is willing to buy/sell for 1 dollar per glass, a investor could buy lemonade from vendor 1 and sell it to vendor 2 while making a 5 cent profit per cup. If both vendors have infinite wealth and supply of lemonade, the system is sustainable and the investor can reap infinite profit. When wealth and supply are constrained, as investors exploit the price difference lemonade supply will build for vendor 2 and cash will grow for vendor 1. The economic laws of supply and demand dictate that the optimal responses for vendors 1 and 2 are to raise and lower prices respectively. As vendor prices converge, the opportunity to profit from the price differential vanishes. In other words, exploitation of arbitrage opportunities diminishes their profitability. Note that in this example with perfect information and no transaction costs the risk experienced by the investor is 0.

Ramping up the arbitrage concept to financial markets, consider the foreign exchange rates in table 2.2.1.

Although the goods are no longer strictly homogeneous, currencies can be directly exchanged and are therefore perfect substitutes in a frictionless market. Due to the

Table 2.2.1 Foreign exchange price matrix

	GBP	USD	EUR
GBP	£1.00	$1.51	€1.11
USD	£0.66	$1.00	€0.76
EUR	£0.90	$1.28	€1.00

discrepancy between GBP price for USD and EUR price for USD an individual with 1 GBP could convert GBP $\rightarrow$ USD $\rightarrow$ EUR $\rightarrow$ GBP and finish with 1.02 GBP or 2% more GBP than their initial investment. As before, this profit is instantaneous, meaning the same trade can be executed repeatedly until the market price difference closes. To reiterate, the market inefficiency will invariably be eliminated because the economic impact of repeatedly executing the profitable arbitrage opportunity effects the currencies' supplies and demands.

Instantaneous profit with no risk is an appealing concept. These types of arbitrage opportunities are exploited by large institutions with below average transaction costs long before the possibility of an individual personally profiting from the inefficiency. Moreover, in most markets, competition among these large institutions to exploit the risk-free opportunities for profit force price differences to be unprofitably small.

More realistic instances of arbitrage are present when a combination of financial derivatives is able to effectively mirror the behaviour of a separate instrument or pool of instruments. If there is any price discrepancy between the portfolios of instruments this is representative of an arbitrage opportunity. It may be difficult to uniquely compose multiple asset pools which perform identically. A more likely scenario is when groups of instruments can be assembled in a fashion which assumes some risk yet yields excess returns[1] for the assumed risk level.

2.2.2 Statistical Arbitrage

Statistical arbitrage is a generalization of arbitrage extending to scenarios which are not entirely risk-free. For example, if transactions may not be executed simultaneously, the foreign exchange arbitrage scenario would technically be considered statistical arbitrage. This is because it is possible for exchange rates to move unfavourably during the process required for exploitation. One market with multiple known statistical arbitrage trading strategies is the equities market. In these

[1]The returns from financial holdings exceeding of those alternative investments with similar risk profiles

markets, the most simplistic of the strategies is known as pairs trading.

Historically speaking, there are assets in equity markets which tend to share a stochastic trend, i.e. move together in terms of price level. A trader might expect this shared stochastic trend between assets with common market sectors, raw inputs, are complimentary goods, or contain various other structural/economic similarities. For example, BP and Chevron spot prices tend to move together because they produce homogeneous products while relying on identical commodity inputs. The similarities such as market demand, crude oil price, and weather conditions are shared between the corporations leading to their parallels in their valuation.

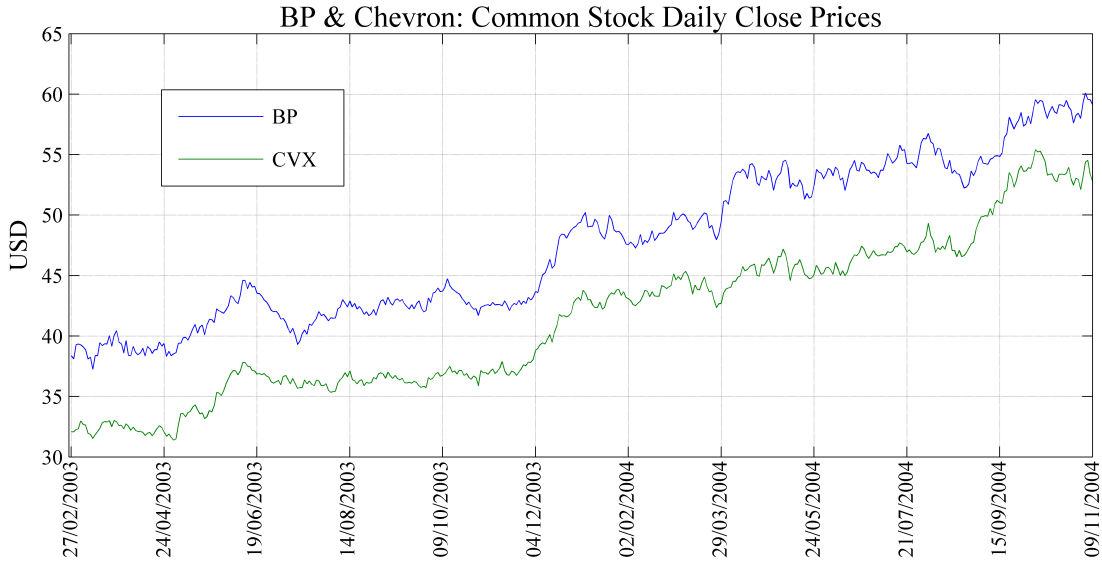


Figure 2.2.1: Background: Close Prices $2003 - 2004$

Given the nonstationary nature of the time series and shared stochastic drift, the selected option for modelling the apparent relationship among the assets is cointegration. Noting that both series pass tests for nonstationarity it is found that the spread between assets is modelled by:

$$0 = (0.9791) * CVX - BP + 7.0314 \tag{2.20}$$

This relationship returns the residual plot in figure 2.2.2. In the residual plot there are a few instances of isolated volatility clustering along with long periods positive serial correlation. The JB normality test returns within the acceptable range leading to

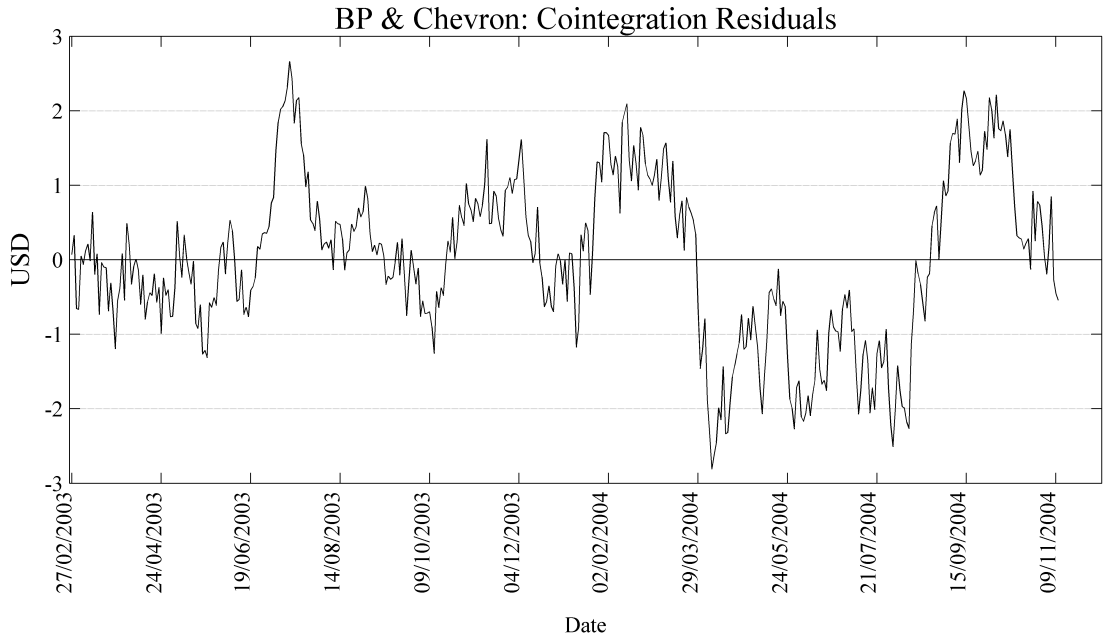


Figure 2.2.2: Background: Cointegration Residuals

the conclusion that the model is well specified.

The trading strategy implemented on this model relates to the mean reverting nature of its residual series. The cointegration model suggests that the spread between the asset pair has a long run relationship which tends to a constant. By adjusting the model by the constant value of 7.0314 the residuals are expected to be a weakly stationary series around 0. The tendency for a 0 spread allows traders to develop expectations of future price movements of the assets based on the current spread level. The expectation of future movements is determined by the current deviation of the spread from 0.

When residual values are large and positive, the mean reverting nature of the spread suggests that in the future, the spread will return to 0. To profit from this expectation, a trader can take a long position in BP stock and a short position in Shell stock. Conversely if the spread is large and negative than a short position in BP and a long position in Chevron covers the expectation of future price movements. When the spread is close to 0, the predictive power of the model also approaches 0. A very simple, in-sample trading strategy using these two assets could take long and short positions with a maximum investment of 100 shares in either asset. Since prior knowledge of the maximum spread value is known, dividing the spread through by the absolute value of its maximum returns a signal series with maximum absolute

value of 1. By then multiplying the signal by 100, corresponding to the maximum number of desired shares in either asset, the signals in figure 2.2.3 for BP and Shell were derived.

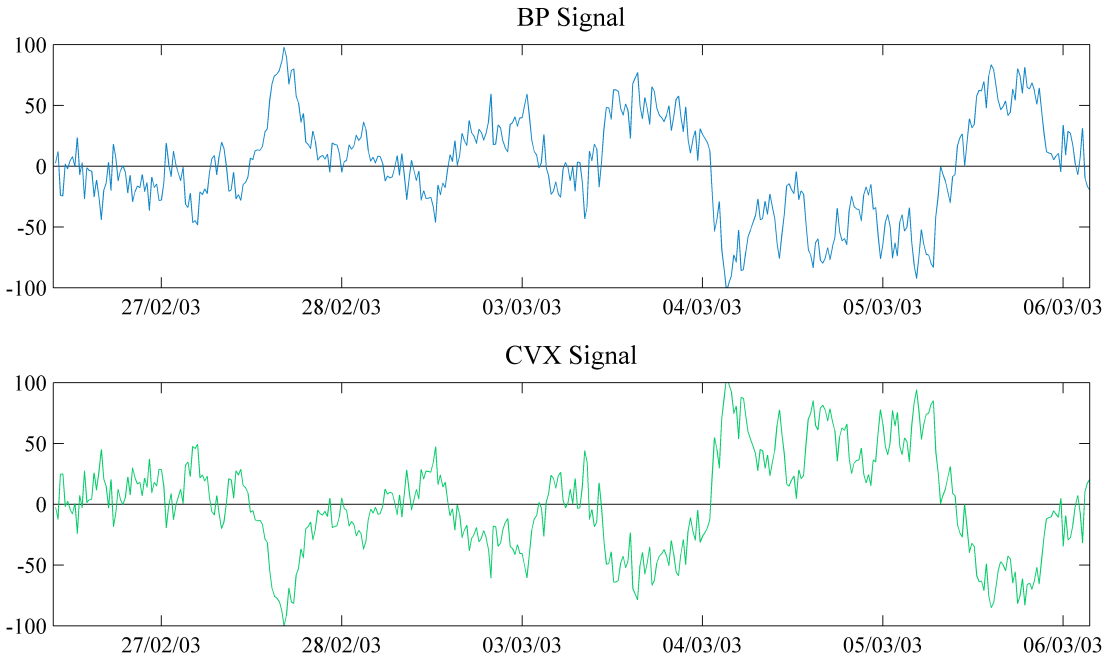


Figure 2.2.3: Background: In-Sample Trading Signals

The signal level indicates the number of shares of each asset to own at each point in time to profit from the strategy. Although the signals are symmetric the magnitude of BP's signal is slightly less than Chevron's because it must be multiplied by its cointegration coefficient of 0.9791. In cases where there is a large difference between the assets' price levels the variation in signal magnitude is more pronounced. Returns of the portfolio can be computed by multiplying each signal by its daily return. This can be seen in figure 2.2.4.

Individually, each asset's return was positive. This is not a necessary condition to meet the expectation of a diminishing spread using the cointegration model. If a trader sees a large positive spread and takes a short/long position in Chevron and BP respectively it is possible for Chevron and BP both to increase in price levels. So long as the increase in Chevron levels is less than that of BP, the large positive spread would have decreased. In that situation, Chevron would incur negative return, and BP would generate positive return. Given a reduction in spread, the returns in one asset

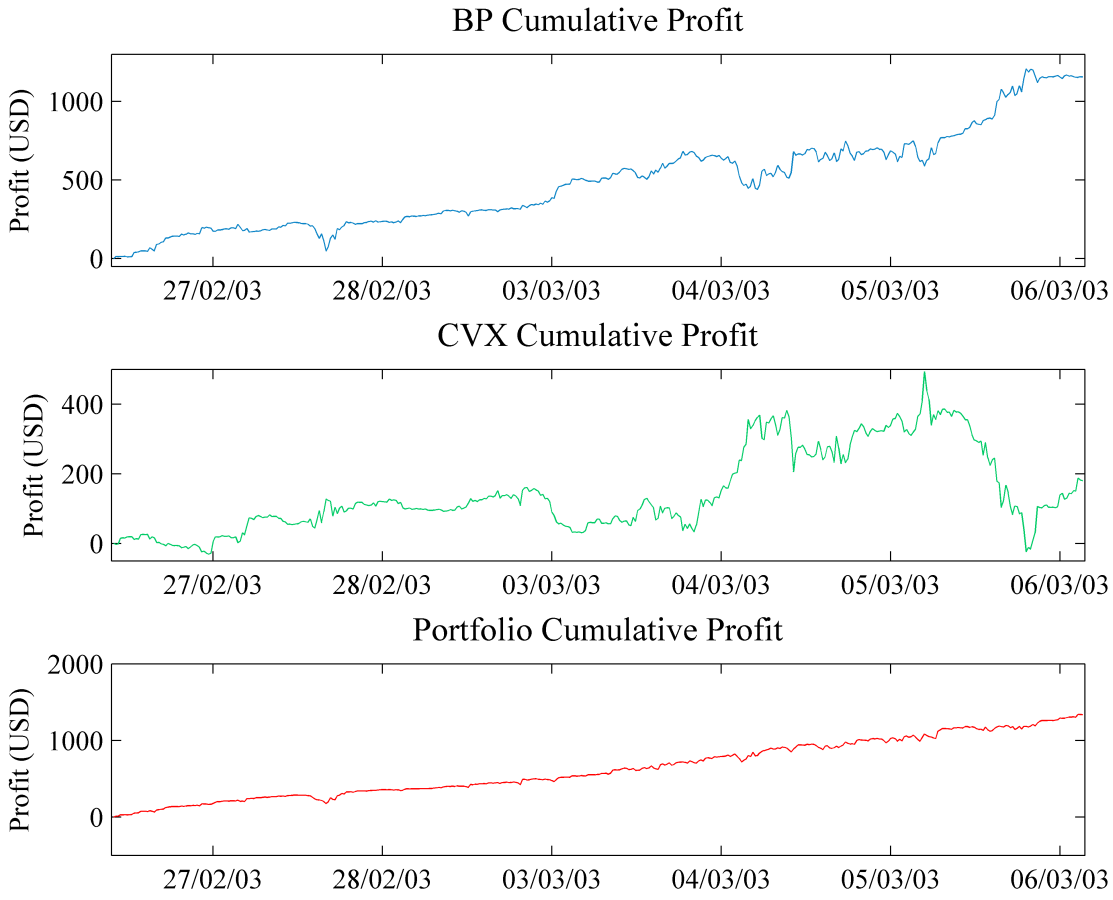


Figure 2.2.4: Background: In-Sample Profit Curves

must always exceed any losses in the other. This means, the portfolio of assets remains positive regardless of the shared directional movements. To recap, when a pair shares a direction of price movement, the losses in one asset are theoretically covered by profit in the other. The implications of this are very significant. If shared movements in price levels do not impact profitability of the portfolio then the portfolio is isolated from global market movements. A portfolio of assets which exhibits this property is known as is market neutral. A market neutral portfolio suggests extremely low risk when models are well specified.

Another key aspect of statistical arbitrage strategies relates to the capital investment. It should be apparent that all long positions are matched with an equal but opposite investment in a short position. This causes the cost realized by investing in long positions to be offset by the short term cash produced by the opposing short position. As such, the cash required to take the opposing positions is approximately equal to transaction costs. This concept is referred to as cash neutrality.

When trade positions have no upfront cash requirement the limitations on exposure to an opportunity are bound by the investor's ability to leverage the position. In practical terms, short positions can be expensive due to transaction costs. Furthermore, financial derivatives such as options and futures present lower cost and greater leverage techniques to utilizing the economically valuable expectations implied by the cointegration model.

For example, purchasing a put option on the asset expected to decrease in value can be a less expensive and a longer term solution than a short. Another distinct advantage to this approach is bounded losses. When entering a long position the maximum an investor can lose is their initial investment. In contrast, a short position has theoretically unbounded loss potential as the asset's price grows. A put option eliminates this unbounded left tail risk since executing the put is not mandatory. The disadvantages associated with put options are also numerous. The fixed strike price of the put in tandem with the fluid decimalized price of a long position suggests than an investor has a maximum cap on their hedge. Options markets for smaller capitalization equities also tend to be illiquid which increases buyer's premiums. Finally, implementing an option based hedge eliminates one of the primary features to pairs trading, its cash neutrality.

Cash neutrality renders some common performance metrics inapplicable. Percentage rate of return, for instance, is undefined when its denominator reaches 0. To compensate, any further discussion of rates of return are measured assuming a

positive cash requirement for both long and short positions.

This adjustment dramatically understates the rate of return achieved for a given investment of cash. To permit comparison with non-cash neutral strategies the Sharpe ratio can be applied [44]. This ratio considers only returns while ignoring cash investment. The ex-post Sharpe ratio is defined in equation 2.21.

$$SharpeRatio = \frac{R_P - R_f}{\sigma_P} \tag{2.21}$$

where

$R_P =$ Return of portfolio investment

$R_f =$ Return of risk-free investment

$\sigma_P =$ Standard deviation of portfolio investment

Sharpe ratio computes the excess returns of an investment strategy over a risk-free strategy and divides the result by the investment strategy's standard deviation of returns. This ratio exhibits desirable properties in measuring efficiency as in increasing function of returns and a decreasing function of return variability. The Sharpe ratio can be measured on any frequency of returns data. Moving forward, all Sharpe ratios are annualized values.

Applying the Sharpe ratio along with the adjusted rate of return metric the model implemented on BP and Chevron common stock is shown in table 2.2.2.

Table 2.2.2 Background: In-Sample Portfolio Annual Performance

Metric	**Value**
Sharpe Ratio	3.010
Mean Rate of Return	21.3%
Mean Nominal Return	$783.31

The reason for the excessive returns is due to the concision of the cointegration period's training data, and the statistical arbitrage strategy's evaluation period. In other words, the trading was performed in-sample with interpolations made from future knowledge of the market. In fact, the only position containing no future information is the last period of the 431 trading days which coincides with November 10^{th} 2004. From this date onward the strategy would be out-of-sample. An extended out-of-sample performance of the strategy can be seen in figure 2.2.5.

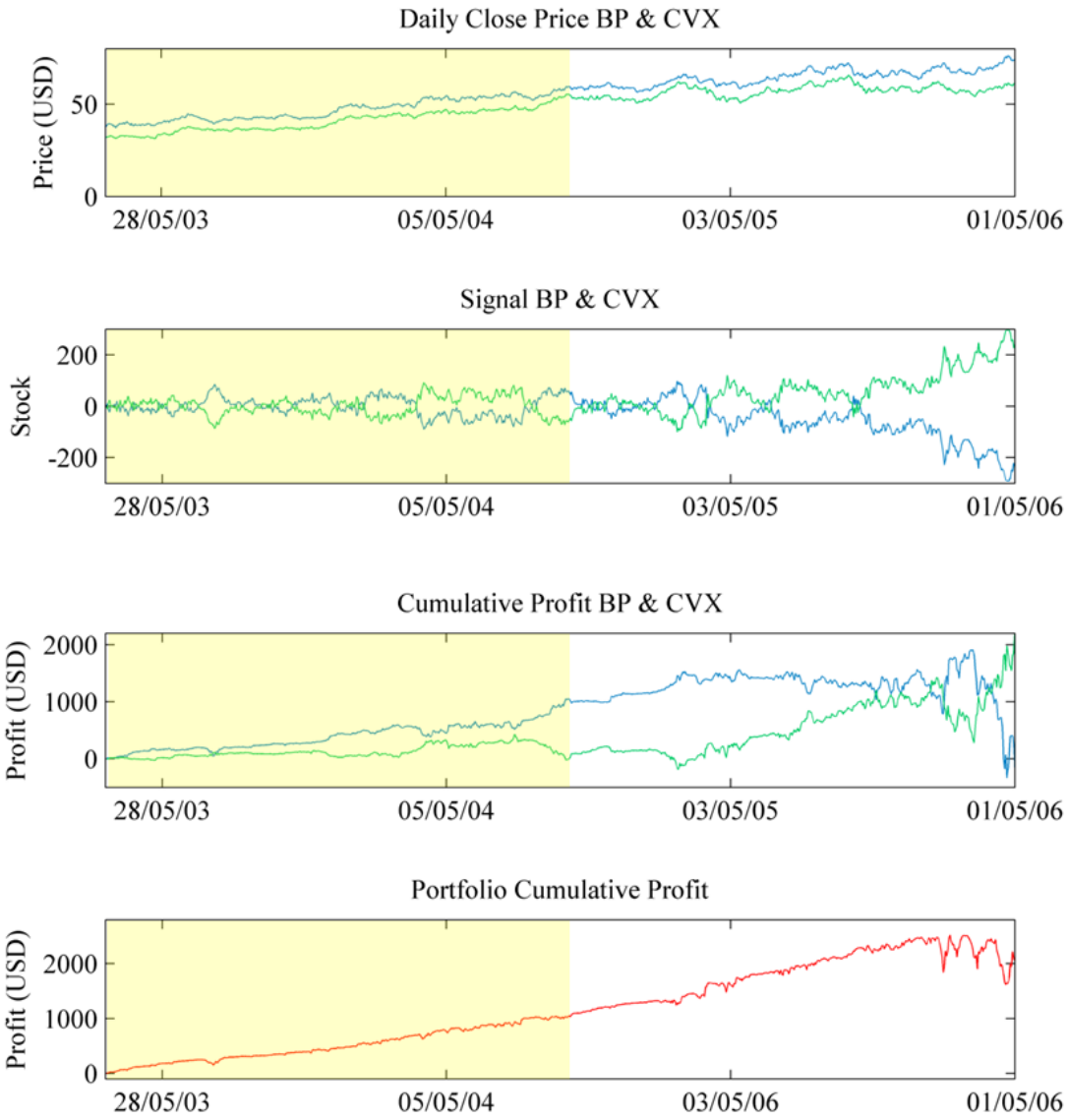


Figure 2.2.5: Background: Out-of-Sample Profit Curves

The yellow section corresponds to the in-sample period. Beyond this point the 'Portfolio Profit' curve illustrates that the evaluation period has a net positive return. The volatility in individual asset performance as well as portfolio performances increases through time from the exit of the in-sample period. This volatility steadily grows and eventually overcomes predictive power of the model rendering the strategy unprofitable.

Table 2.2.3 Background: Out-of-Sample Annual Portfolio Performance

Metric	**Value**
Sharpe Ratio	0.657
Mean Rate of Return	7.9%
Mean Nominal Return	$640.04

As expected, the performance metrics all decreased relative to in-sample figures. Equity pairs which track as well as the example are rare. There are few industries dominated by a small set of known factors which tend to induce the shown level of similarity between independently managed companies.

As computational resources become less expensive and statistical arbitrage strategies more popular, instances of trading opportunities become a scarcity. Identical to true arbitrage, exploitation of statistical arbitrage strategies creates market forces that adversely impact their own profitability by altering market supply and demand.

The example trading strategy introduced in this chapter was built for demonstration purposes only. With increasing efficiency of financial markets the developed simplistic model would be unlikely to yield profit in a modern market. One clear example of the model's shortcomings is the exclusion of transaction costs.

More sophisticated traders may attempt to develop custom sets of assets which tend to be cointegrated in the aggregate by trading baskets of assets. Alternatively, using the Johansen framework[46] traders may develop cointegration models for a subset of a market sector by defining a relationship among several assets. The primary issue in this approach is the risk of over fitting the model when the number of traded assets is large.

In summary, the 3 properties discussed which make statistical arbitrage such an attractive investment type are, positive expected return, market neutrality, and minimal cash requirement. The sophistication of the trading algorithms implemented

are limited only by investor's perceived ability to maintain these attributes. Furthermore, the name statistical arbitrage need not be constrained to expectations developed on cointegration models. Any model capable of producing reliable market expectations aiding in trading can be viewed as statistical arbitrage.

The pairs trading statistical arbitrage scenario is extensively discussed because it presents a fundamental building block of the network model and a validation. A cointegration model among a small number of assets aims to create a long-run associative network of asset interactions. The network model aims to perform the same task on the entire market. Similarly, pairs trading utilizes a cointegration model to create an algorithmic trading strategy. One validation of the network model's accuracy is an algorithmic trading strategy which bares some similarity to pairs trading.

As with all trading strategies, the quality of a statistical arbitrage system can be measured only through its out-of-sample profitability in live market conditions. Without proper testing this may be a risky prospect. To further validate the expected source of profitability prior to introducing risk it can be beneficial to observe the strategy in an environment where inner-workings are known. For this task a simulated or artificial market can be applied.

2.3 Artificial Markets

Artificial markets were necessary to validate contained research as a mechanism for overcoming the opacity problem in true financial markets. In development of the network model, expectations were made regarding market microstructure and causes of asset price movements which are not directly observable in real financial markets. In order to tie realized price movements back to information shocks, uncover mis-pricing, and manage risk we created an artificial market with full transparency. In the artificial market a set of arbitrary factors were partially determinant of artificial asset prices. Perfect information of these asset compositions was retained throughout the process. The network model was then given access to the artificial market and relationships among the artificial assets were modelled. By then deconstructing the assets to their known factor components the model's accuracy was directly assessed. This task would have been impossible in a true market.

Development of artificial markets is an emerging trend which attempts to model or simulate behaviour of market dynamics through interactions of elements in a complex

system. Due to lack of information, computational resources, and understanding of how markets function, much work has attempted to model emergent behaviour of markets as opposed to markets themselves. Modern financial markets are complex adaptive systems with millions of interacting agents. Given that all of these agents have differing goals, expectations, and information, it is not surprising that there is no perfect artificial market for testing research. Instead, there are many forms of artificial markets each with a target audience aiming to replicate a small fraction of market microstructure. Our review of these artificial markets falls under 3 areas of study.

1. Systemic Risk
2. Algorithmic Trading
3. Policy Making

Despite most work being coupled with an intended application, some articles are written purely to propose guidelines in producing artificial markets for computational research [76]. The most widely accepted models are agent-based. In this paradigm assets are priced by the interaction of agents as opposed to pre-determined or directly stochastic processes. Sophisticated models can take into account other sources of market movements for example company decisions and market externalities [5].

2.3.1 Systemic Risk

Artificial markets need not be limited to agent-based stock models. A growing use can be found in stress testing banking networks. Following the 2008 banking crisis the concepts of systemic risk and financial contagion increasingly have become areas of study. Traditional methods of estimating banks' exposures to financial downturns such as value-at-risk were demonstrated to fail in capturing risk of a domino effect associated with inter-bank lending agreements. Systemic risk is the danger when interrelationships in a system, or network, allow shocks introduced to a portion of that system to be transferred across its breadth. The more interconnected the system, the greater the danger of negative shocks taking out nodes in the network.

Using historical data drawn from the Mexican banking system, Canedo and Jaramillo [14] developed a technique for measuring the stability of interbank lending exposures. The chosen metric to determine interconnectivity of the simulated banking systems was taken to be the matrix of interbank lending. By representing

each bank in the system as a node in a network, any bank which lent funds to other banks in the system became linked to the recipient. Should the receiving bank, 'Bank A', default on their obligations, all banks which had granted Bank A a loan would be forced to write off the receivable pushing themselves further toward default. In extremely interconnected environments a system which was not obviously unstable could prove problematic when accounting for 2^{nd} and 3^{rd} order effects as a cascade of lenders became insolvent.

In this example a network model of systemic risk was designed to incorporate two elements.

1. Initial random shock

2. Financial contagion

It was assumed that initial random shocks were created by separate and independent failure probabilities for each institution. These probabilities of failure were denoted p_i where each i was a unique bank. One time step beyond the random shock phase the propagation of initial failures through the banking sector was found by the matrix of interbank exposures designated $d_{i,j}$ representing bank i's exposure to bank j. The total loss function $L()$ resulting from any given initial shock for failed bank set F was defined as:

$$L(F) = \sum_{i \in F} l_i + \sum_{i \in C(F)} l_i \tag{2.22}$$

In other words, the total losses associated with initial bank failure set F was the sum of all initial failures and those associated with the following financial contagion $C(F)$. Membership in set $C(F)$ at any time instance k was determined for bank i by solving for classifier θ_i^k.

$$\theta_i^k = \begin{cases} 1 & if \sum_{j \in D^k} d_{i,j} < u_i^k \\ 0 & otherwise \end{cases} \tag{2.23}$$

Where u_i^k was the nominal loss figure for bank i to remain solvent

When the probabilities of each banks' failure due to initial shock, p_i, thresholds for default, u_i^k, and interbank exposures, $d_{i,j}$, were known, all combinations of initial bank failures F, with resulting total losses $L(F)$ could be assembled into a probability

distribution of losses for the system. If the probability of significant losses was high the system was taken to be unstable. Similarly, if the probability of significant losses was low then the system was deemed stable. In trivial cases with few institutions it is possible to compute the probability distribution deterministically. The problem of computing all possible initial failures F grows exponentially with the number of agents in the system. As a result, in non-trivial cases the probability distribution may be estimated via Monte Carlo methods.

2.3.2 Algorithmic Trading

Use of artificial markets in finance is growing. This trend is particularly apparent when simulating or back-testing real world algorithmic trading strategies. The need for artificial markets arises from the inability for traders to effectively simulate the impact their trading strategy has on the market at large. For example, placing a large bulk buy order on a single common stock has the potential to execute a significant portion of that asset's sell side limit order book. As the buy order matches offers in the book the current market price tends to rise. The complex multi-agent nature of financial markets makes it difficult to determine how other market agents would react to the simulated trade. As a result, any interaction between a simulated agent and a real financial market renders future values in that market inaccurate. Artificial markets are one of the tools researched to help better understand the impact algorithmic trades may have on market dynamics.

Algorithmic traders and computational scientists have taken many approaches to address the problem by creating artificial or simulated market environments in which to test their algorithms. Many researchers develop a custom artificial market appropriate for their test problem but others have created generic models exhibiting the features most frequently required in commonly researched areas. Among the general purpose models is Hammel and Paul's multi-agent trader artificial market [32]. In their market all agents applied different noise-trading strategies. It was demonstrated that interactions among agents produced the appearance of market patterns similar to those visible in real markets. This market exhibited a time crowding behaviour of market participants' expectations and a dampened positive feedback loop of market volatility. Other instances of agent based artificial markets have incorporated heterogeneous trading strategies [6]. The extent to which different strategies were able to outperform the market and each other was taken as a measure

of market efficiency. In an efficient market, it was assumed that no excess returns would be possible. Therefore, differing returns among the strategies indicated a breakdown in the efficient market hypothesis[2].

Artificial markets with agent based algorithmic trading can also be used to show the interdependence of agents within financial markets. Izumi, Toriumi and Matsui [41] demonstrated that in agent based models allowing competition of unique trading strategies each agent was impacted by both their own strategic choices and their competitors' choices. By allowing the agents to create trades under multiple simulated market conditions it was determined that the best trading strategy in each environment depended both on its own parameters and those of other agents.

2.3.3 Policy Making

Policy makers find themselves in a similar position to algorithmic traders in that the impact of any decisions they choose to implement changes market dynamics and renders use of historic data inappropriate. To compensate, artificial markets have become the natural proving ground for impending policy decisions.

Efforts surrounding monetary and fiscal policy decisions are a common source of research. These day-to-day decisions allow policy makers to provide a guiding hand in market direction and volatility. While loose expectations are fairly intuitive, developing expectations about specific impacts or magnitudes proves difficult. Agent based models allow research into expected impact of decisions such as interest rate changes or governmental spending on variables such as market volatility, unemployment levels, income distribution, and growth projections [24]. Similar work also utilizes agent based models to estimate impacts of policy decisions on the business cycle [65].

Comparable artificial markets have also been exploited to test major regime changing policy decisions. Yeh and Yang [75] developed an artificial market to test the feasibility of price limits as an alternative to halted trading during periods of market instability. The use of an artificial market enabled them to estimate impacts on market volatility, price distortion, and liquidity. These methods were also considered when talks of introducing a Tobin-like tax[3] were under review. Mannaro, Marchesi,

[2] A financial theory stating that efficient markets are characterized by perfect or near perfect information of participants. A significant implication is the resulting inability for any investor to consistently achieve risk-adjusted excess returns for any trading strategy.

[3] A tax on financial transactions

and Setzu raised concerns from their heterogeneous artificial market that a Tobin-like tax could reduce trading volumes and increase market volatility[59].

2.4 Modern Portfolio Theory

Mean-variance analysis or MVA is a tool aiding in optimal asset allocation for financial portfolios [60]. Under strict assumptions MVA solves for asset weights which, when combined, return the Pareto front trade-off curve between portfolio's riskiness and return. The contribution of this tool was a straightforward way of measuring the diversification benefits obtained through holding multiple assets. Portfolios of assets which are strongly correlated in their returns suggests a poor diversification and oppositely, multiple assets with weakly, or negatively, correlated returns indicates significant diversification. When asset returns are uncorrelated, losses in a single asset are linearly unrelated to returns in other held assets. Assuming all assets have a positive expected returns, minimizing correlation among held assets' returns reduces the variance of the portfolio's returns. In this case, variance of portfolio returns is used as a proxy for riskiness.

Although the concept fits in a computational finance context we do not use MVA as a portfolio diversification tool. Instead we use mean-variance analysis as a signal processing technique to diversify the source of predictability for a set of signals. Stripping away the financial application, mean-variance analysis computes the best possible trade-off curve between one independently combining metric and another which diversifies according to correlation. The tool was viewed in this fashion and adapted to enable recombination of sensor data output by the network model. The data were pre-processed into simulated expected returns and expected risks. The resulting values were entered directly into MVA. The output produced a reduced number of higher quality sensors which were linearly diversified 'portfolios' of low quality input sensors.

The theory of mean-variance analysis is based around a one period model. Expected return of asset a is represented by the historic return of a.

$$E(R_{a,t}) = R_{a,t-1} \tag{2.24}$$

where

$E() =$ Expectation function

$R_{a,t}$ = Returns of a
$R_{a,t-1}$ = Previous period returns of a

and

$$R_a = R_{a,t} \tag{2.25}$$

Historic variance of asset A is taken to be its risk level. Again, historic variance of a's return is assumed to be a perfect predictor of future expected variance.

Using this framework for risk and return it is possible for investors to make asset selection decisions for one-asset portfolios. Consider figure 2.4.1.

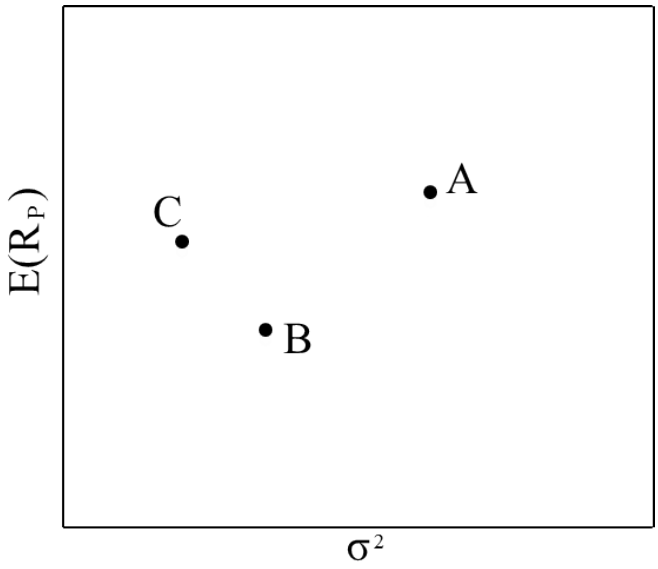


Figure 2.4.1: Background: Assets A,B and C in Risk/Return Space

Take A, B and C to be assets with risks and returns as plotted in Figure 2.4.1. Notice the trade-off between assets A and B such that $E(R_A) > E(R_B)$ and $E(\sigma_A^2) > E(\sigma_B^2)$. For this reason, neither A nor B strictly dominate each other. The choice between A and B becomes a matter of risk aversion level in investors. An extremely risk averse investor would select B, while a less risk averse investor would select A. In contrast

$E(R_C) > E(R_B)$ and $E(\sigma_C^2) < E(\sigma_B^2)$ thus C dominates B unconditionally. Any risk averse investor, regardless of degree would prefer asset C to asset B.

As mean-variance analysis is intended to quantify diversification benefits, expected return and expected risk of individual assets must be generalized to multi-asset portfolios. There is an assumption of no relationship among expected returns in portfolios. This means the expected return for a portfolio is simply the sum of expected returns for each asset weighted by proportionate allocation in the portfolio.

$$E(R_P) = \sum_{i=1}^{n} (w_i E(R_i)) \tag{2.26}$$

where
$E(R_P)$ = Expected return of the portfolio
$E(R_i)$ = Expected return of i^{th} asset the portfolio
w_i = % Allocation weight of i^{th} asset in the portfolio
and

$$\sum_{i=1}^{n} w_i = 1 \tag{2.27}$$

Variance of a portfolio is not as straight forward. Mean-variance analysis discounts overall portfolio variance by taking into account the possibility that assets may move with differing magnitudes or signs. The linear metric of Pearson's correlation is used to estimate the required relationship among assets.

$$\sigma_P^2 = \sum_{k=1}^{n} w_k^2 \sigma_k^2 + \sum_{k=1}^{n} \sum_{\substack{i=1 \\ i \neq k}}^{n} w_k w_i \sigma_k \sigma_i \rho_{ik} \tag{2.28}$$

where
ρ_{ik} = Correlation between returns of i^{th} and k^{th} asset

Thus, all else being equal, applying larger weights to minimally correlated assets yields greater diversification benefits leading to reduced portfolio return variances.

Suppose 2 assets of known expected return and risk were weighted to construct a portfolio. If $\rho_{AB} = 1$ then the assets would to be perfectly positively correlated. This implies no possibility of risk avoidance through diversification. Figure 2.4.2 shows the

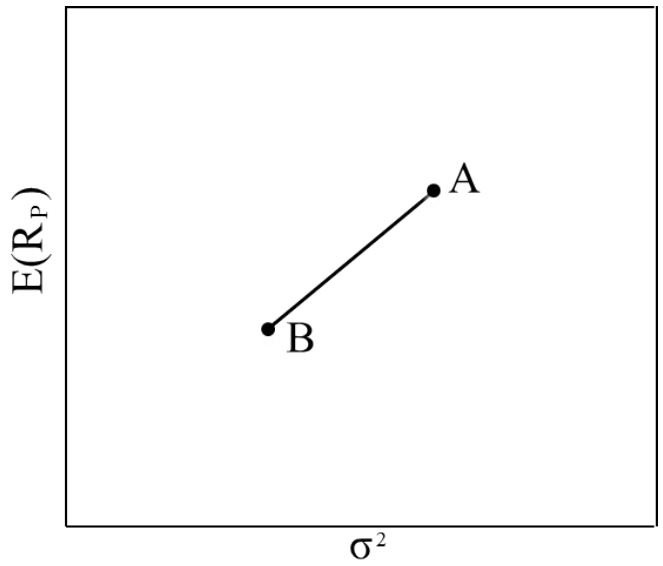


Figure 2.4.2: Background: Assets A and B Portfolios $\rho_{A,B} = 1$

attainable trade-off curve between assets A and B in risk/return space.

Provided $\rho_{AB \neq 1}$ the attainable profit curve would not be a straight line between the assets. In the most extreme example, take $\rho_{AB} = -1$. In this case, the portfolio set contains a portfolio in which expected return is between those of A and B with a risk of 0. Since both A and B have positive expected returns the risk-free portfolio generates positive expected returns. This relationship is shown in figure 2.4.3.

When $1 > \rho_{AB} > -1$ the possible portfolio curve more closely resembles figure 2.4.4. In this scenario, some diversification is possible but complete risk mitigation is not an option of the portfolio set. The critical point marked in red is of particular interest to investors. Given the MVA risk, return framework any point below the critical value represents a dominated portfolio. Any portfolio above the critical point is considered 'efficient'.

When the number of assets made available in portfolio selection increases, the number of portfolio allocation curves grows exponentially. Continuing with the visualizations, an aggregation of all portfolio allocation curves with large number of assets may appear as in figure 2.4.5.

The representation of portfolios in risk versus expected return space allows for equitable expected performance comparisons regardless of the number of assets per

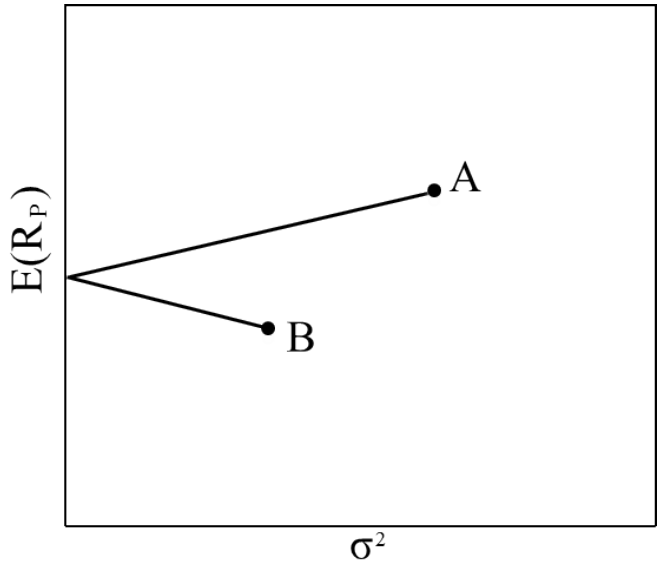


Figure 2.4.3: Background: Assets A and B Portfolios $\rho_{A,B} = -1$

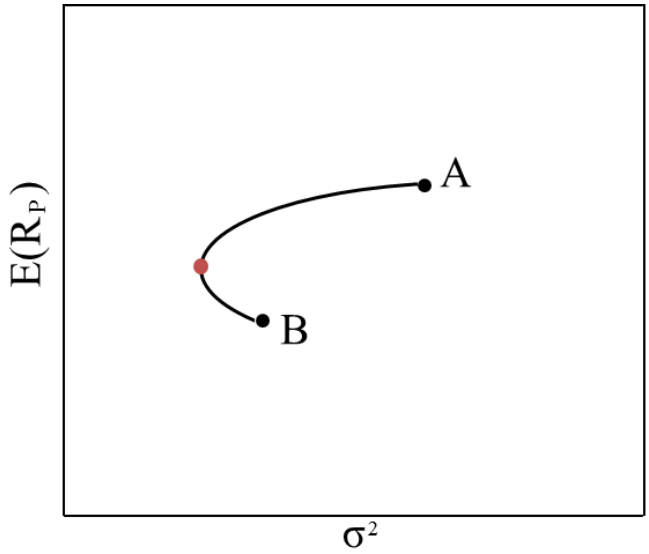


Figure 2.4.4: Background: Assets A and B Portfolios $1 > \rho_{A,B} > -1$

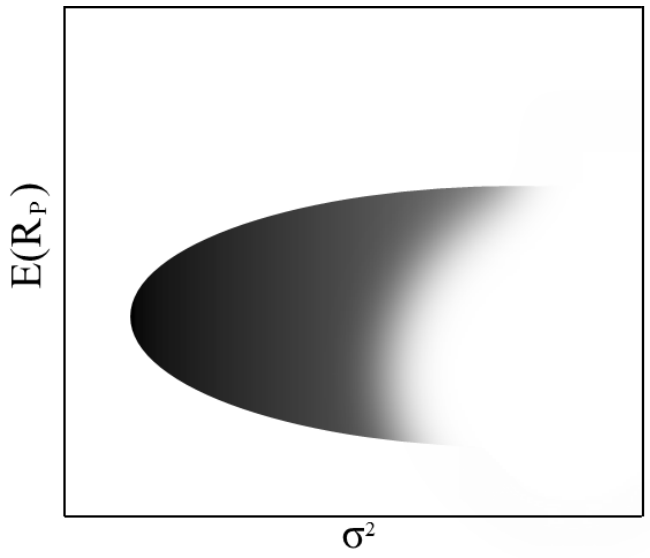


Figure 2.4.5: Background: Multi-asset Portfolios

portfolio. The set of portfolios with highest expected return for a given level of risk, or inversely, lowest risk for a given level of return, are known as the efficient frontier. The efficient frontier can also be thought of as the set of portfolios any rational, risk adverse investor would choose from.

2.5 Genetic Algorithms

Genetic algorithms or GAs are population based metaheuristics for optimisation problems commonly utilized where deterministic solutions are unknown or prohibitively computationally expensive. These optimisation problems are often characterized by an extremely large and discrete search spaces [63].

2.5.1 Single Objective

Single objective genetic algorithms or SOPs are optimisation problems where there is only one fitness or cost function to be maximized or minimized respectively. A basic genetic algorithm undergoes 4 phases. Those phases are:

1. Initialization
2. Selection
3. Recombination

4. Mutation

The algorithm begins by initializing a 'population' of N candidate solutions or 'individuals' for the current optimisation problem. Each individual is has a 'genome' which encodes inputs to the function being optimised. For example:

$$F_n = \sum_{i=1}^{5} g_{n,i} \tag{2.29}$$

where

$g_{n,i}$ = the i^{th} gene for the n^{th} solution candidate's genome

F_n = fitness value of individual n in the population of size N

For this problem, each position or 'gene' in the genome can contain a value of 0 to 9. Clearly this simple function is maximized with a value of 9 at every gene. To achieve this result programmatically a genetic algorithm applies an iterative process. The iterative process 'evolves' a population of candidate solutions which converge to a global or local optima.

When discussing genetic algorithms the jargon often revolves around the trade-off between exploration and exploitation. That is, exploration of potentially valuable gene arrangements, and exploitation of gene arrangements already known to yield above average fitness. In the following sections factors impacting the exploration to exploitation trade-offs are noted.

Initialization

During the initialization phase the user must determine an encoding and decoding mechanism to convert strings of characters, digits, etc. to viable inputs for the optimisation problem or fitness function. The fitness function is the sole source of each algorithm's perception as to a solution candidate's quality. For example, in the case of optimising the previously presented fitness function, a genome of $[3, 1, 6, 3, 2]$ evaluates to 15. Another genome of $[9, 6, 8, 7, 2] = 32$. From this result the second genome is thought to be superior to the first.

Common encoding types include boolean/binary, integer, and permutation. Initial populations are seeded randomly. An example initial population of each type where the number of individuals per population, N, is 10 and the number of genes in a

genome, G, is 5 can be found in tables 2.5.1 to 2.5.3. The populations are arranged in an N by G structure.

Table 2.5.1 Background: Boolean Initial Population: N = 10, G = 5

0	1	1	1	0
0	0	1	0	1
0	1	0	0	0
1	0	1	0	0
1	1	1	0	1
0	0	0	0	0
1	1	0	1	1
0	1	1	0	1
0	1	0	0	0
1	0	0	1	0

Boolean encoding may be used where each digit toggles use of some external feature. Alternately, some users prefer to express more complicated encodings in binary to retain use of the many compatible choices for recombining and mutating the type.

Table 2.5.2 Background: Integer Initial Population: N = 10, G = 5

8	5	3	7	0
2	6	4	0	4
7	6	2	6	9
2	6	1	0	8
8	1	0	6	4
4	0	3	4	4
6	4	0	3	0
2	0	6	1	5
6	9	1	1	4
1	4	3	5	8

The example given in maximizing the sum of the genome applied integer encoding. Integers are valuable when multiple variables for the target fitness function lie in a discrete, known, range as in figure 2.5.2.

Permutations are the final common genome encoding method discussed. This type can be useful for scheduling and routing problems, a common application of genetic

Table 2.5.3 Background: Permutation Initial Population: N = 10, G = 5

5	1	4	2	3
2	5	4	1	3
4	5	2	3	1
2	1	5	4	3
4	3	5	2	1
2	5	3	1	4
1	2	3	5	4
5	1	3	4	2
2	4	3	1	5
1	3	5	4	2

algorithms.

Selection

Selection is the process responsible for the gradual improvement of population fitness. In the selection phase a number of candidates solutions are chosen from the current population, P_t, to be recombined and produce a new candidate solution in the next 'generation' of candidate solutions, P_{t+1}. By repeating this process N times, a sufficient number of candidate solutions are selected in P_t for recombination which yields a P_{t+1} of equal size. The method of parent selection is one of the most crucial decisions when writing a GA. While there are many variants of selection techniques, roulette wheel and tournament are both widely acknowledged with well known properties.

Roulette wheel selection gives each solution in the population a percentage chance of selection equal to its proportionate fitness in the population. For example, if there are 5 individuals in a population with fitness values $[1.8, 2.4, 1.2, 9.3, 4.6]$ the first individual will be chosen an average of $1.8/(1.8 + 2.4 + 1.2 + 9.3 + 4.6) = 9.33\%$ of the time. The percentage of selections for all individuals are $[9.33\%, 12.44\%, 6.22\%, 48.19\%, 23.83\%]$. Notice individual 4 is selected 48.19%. Roulette wheel selection is not robust to individuals significantly exceeding the population average fitness. Any individual which does, e.g., individual 4, may be prematurely overrepresented in the P_{t+1}. In other words, roulette wheel selection may excessively exploit above average genomes to the detriment of exploring other possibilities. The primary advantage of this form of selection is the lack of parameters. A well parametrized tournament selection can control the balance of

exploration v.s. exploitation more carefully at the cost of introducing variables.

Tournament selection begins by randomly selecting h individuals from the population to compete in a tournament. In the tournament the individual with the highest fitness is selected. When the process is repeated 2N times, the tournament winners can be grouped for recombination. The parameter h controls the balance of exploration versus exploitation. As h increases, exploitation grows while exploration shrinks. As a practical demonstration, review the roulette wheel selection example. Applying the tournament selection procedure with h of 2 the probability of individual 4 being selected is the probability of it being chosen for the tournament multiplied by the probability of the individual winning in any given tournament or $(8/20) * 1 = 40\%$. Increasing h to 3, 4, and 5 the percentage selection increases to 60%, 80% and 100% respectively. Given a realistic population size the maximum probability of selection for the best individual in the population can be adjusted in smaller increments of size $1/N$.

Recombination

Recombination is the process of combining two or more parents selected in P_t to produce a candidate solution for P_{t+1}. Unlike the various forms of selection which are universally applicable to all encoding types, not all recombination types are compatible with all encodings. In the same way that selection is compared in terms of exploration versus exploitation, recombination can be thought of as varying in genome destruction.

Destruction refers to spiting adjacent runs of genes in parent genomes. Depending on fitness landscape, the cause of a genome's above average fitness can be attributed to short 'runs' of beneficial genes. If the run is preserved in the offspring's genome, the offspring's fitness tends to outperform a comparable offspring without the run.

The uniform crossover type is primarily intended for problems where all elements in the genome are independent and relative location of genes to each other has no baring on fitness. Uniform crossover creates a new candidate solution by randomly selecting genes from 2 or more selection winners. Genes in the random selection must be drawn from the equivalent gene location from any parent. Uniform crossover is the most destructive of the approaches covered.

One-point crossover combines parent genomes at a single randomly selected location gene. All genes prior to this point come from parent 1 and all genes following

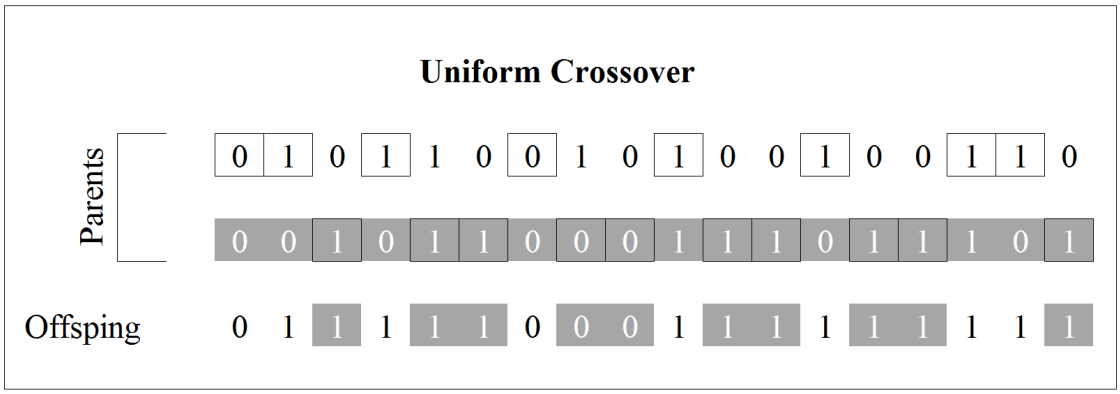


Figure 2.5.1: Background: Uniform Crossover

it are from parent 2. This approach preserves a significant number of gene runs from both parents and is less destructive than uniform crossover. Reducing run destruction benefits offspring genomes in problems where adjacency of genes is at least partially determinant of genome fitness. An important consideration in one point crossover is that the further apart genes are, e.g., $g_{n,1}$ to $g_{n,5}$ is further than $g_{n,1}$ to $g_{n,3}$, the less likely they will both be present in the offspring genome. Therefore, it may be difficult to identify beneficial structure among genes which are widely separated.

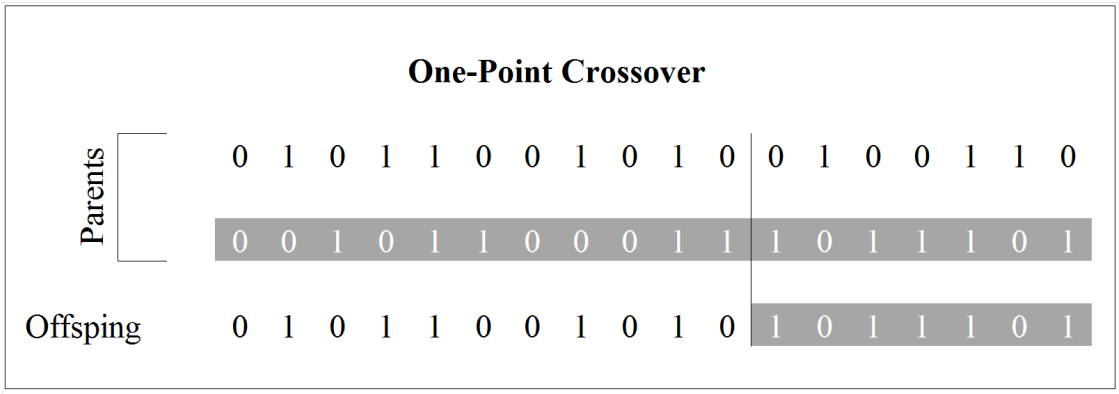


Figure 2.5.2: Background: One-Point Crossover

Two-point crossover operates much in the same fashion as one-point crossover. The primary difference is that end points of the genome are not forced to be the end points of crossover which reduces the issue of structure among distant genes. This type of crossover is appropriate when horizontal genome transcription is acceptable to the problem type. While two-point crossover is less destructive to gene runs than

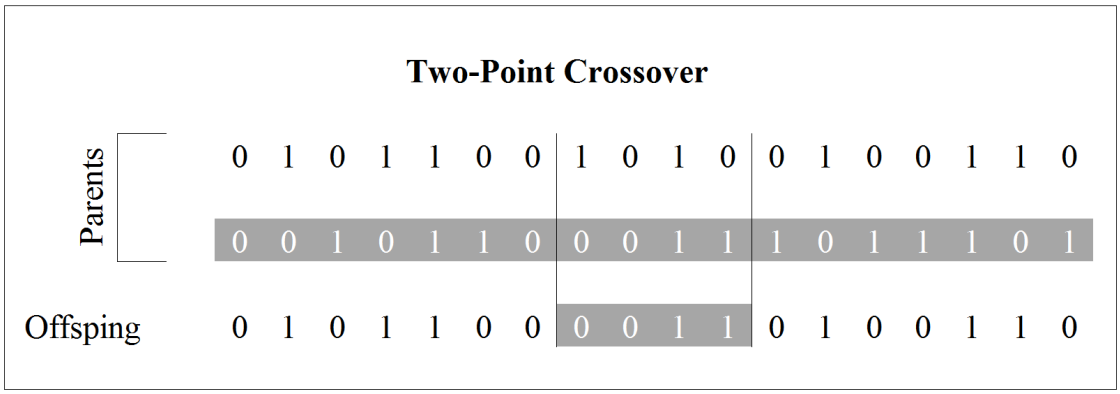


Figure 2.5.3: Background: Two-Point Crossover

uniform crossover, it is more destructive than one-point crossover.

Deciding which recombination type will be most effective is a difficult task. Expert knowledge of the optimisation problem and encoding scheme can be helpful. Further information on recombination types can be found in [63].

Mutation

The purpose of mutation is to reintroduce or maintain a diverse gene distribution in the population. In early generations of the genetic algorithm it is possible for exploitation of found structure to prematurely reduce the exploration of certain genes. Mutation is the operator to help alleviate the issue. Similar to recombination, encoding limits compatibility of various mutation types. Despite this, mutation for most non-permutation encodings adheres to the same set of principals. First, each gene is iterated over and a random number is selected. If the random number falls within a predefined activation range then a mutation is inserted. If the random number does not fall in range, the iterator moves to the next gene location. Boolean mutation simply flips the relevant bit when the mutation function is activated. Integer mutation randomly selects a new value from the originally accepted range in the initialization phase. A simple approach for permutations is to randomly switch two locations on the genome to preserve the validity of the candidate solution.

2.5.2 Explanation

By repeating the phases of evaluation through mutation the population of solution candidates' fitness values tend to improve. Upon meeting an exit criteria the

repetition is stopped and the best solution candidate is chosen as the solution to the optimisation problem. Two similar explanations of why genetic algorithms exhibit this optimising behaviour are embodied in the building block hypothesis [28] and schema theory [33]. The building block hypothesis states that highly fit solution candidates are composed of short runs of genetic material generally associated with higher fitness values. When high fitness individuals are chosen, these building block have a tendency to persist, grow, and combine with those from other high fitness individuals. By repeating the process, the genetic algorithm provides loose guidance to an otherwise random search. Schema theory draws much the same conclusion stating that low-order schemata with high fitness grow exponentially in representation throughout the population.

2.5.3 Multi-objective

Multi-objective optimisation problems exist where more than one criteria must be simultaneously optimised. In these cases, any solution which is not strictly dominated along all criterion axes is considered to be on the best known Pareto front. With each additional fitness criterion the complexity of the problem grows significantly. Genetic algorithms as described in the previous section are poorly suited for these types of problems because they aim to converge the population of candidates to a single solution. In contrast, a genetic algorithm designed for multi-objective optimisation must converge to the best possible Pareto front which may contain an infinite number of solutions. While there is an entire field of genetic algorithm research devoted to this area of study we review only the 3 most academically vetted algorithms. These algorithms are PAES [49], SPEA II [81] and NSGA II [22].

The intended application of the multi-objective genetic algorithm was to be used as a single period portfolio recalibration engine in validation of the network model. As mentioned in the introduction chapter, the portfolio diversification task for the algorithmic trading validation was adapted into a computationally complex yet directly measurable optimisation problem. The performance criteria were selected to minimize the portfolio's risk profile and maximize expected future returns. To solve this problem for each back-testing period a custom multi-objective GA was designed. Given that the back-testing period contained $+2,000$ periods, execution time was a major consideration when selecting a MOGA for adaptation.

PAES

PAES or Pareto Archived Evolution Strategy is an iterative local-search genetic algorithm [49]. The theory behind PAES is to employ local searches surrounding solution candidates currently existing on best known Pareto front guided by information in the entire population. Though there are 3 originally proposed variants we focus on the most basic denoted $(1 + 1)$-PAES. The structure is as follows:

Algorithm 1 Multi-objective GA: (1+1) - PAES Process

procedure ProcessPopulation
  $\bar{P} \leftarrow$ Archive of solution candidates
  $\bar{N} \leftarrow$ Maximum number of solutions in archive
  $s \leftarrow$ Solution candidate
top:
  $s =$ **randomly initialized solution candidate**
  insert copy of s **into** $\bar{P}$
  while $ExitCondition$ **is** $False$ **do**
    m = **mutate** s
    if m **strictly dominates** s **then**
      $s = m$
      insert s **into** $\bar{P}$
    else
      if no solutions in P **strictly dominate** m **then**
        Apply archive update function $update(s, m, \bar{P})$

The update function is another logic tree with dependencies including the density distribution of archived solution candidates along the Pareto front and if the archive is full. Knowing that one goal of PAES is to provide the widest possible range of nondominated solutions, candidates seeking membership to sparse areas of the archive are given preference over solutions in highly crowded areas. The update function can be viewed in algorithm 2.

PAES was originally developed as an approach to the off-line routing problem and highly emphasized simplicity. Due to this simplicity the authors focus on its use as a potential benchmark for other multi-objective genetic algorithms.

In comparison to the previously outlined structure of single objective genetic algorithms the most notable difference of PAES relative to other multi-objective genetic algorithms (MOGAs) is the absence of recombining multiple candidate

Algorithm 2 Multi-objective GA: (1+1) - PAES Process Update Function

```
procedure UPDATE(s,m,P)
    P̄ ← Archive of solution candidates
    N̄ ← Maximum number of solutions in archive
    s ← Solution candidate
    s̈ ← Maximally crowded solution candidate in P̄
top:
    if number of solutions in P̄ < N̄ then
        insert m into P̄
        if m less crowded than s in P̄ then
            s = m
    else
        if m less crowded than s̈ in P̄ then
            swap s̈ with m in P̄
    if m less crowded than s in P̄ then
        s = m
```

solutions. This difference classifies PAES as a local search algorithm.

SPEA II

SPEA II [81] is a revision of Strength Pareto Evolutionary Algorithm or SPEA [80] which iteratively updates an archive of fixed size representing the most recent generation's estimate of the Pareto optimal frontier. Improvements over SPEA include a more comprehensive view of domination and a solution candidate spacial density estimate (SDE). The SDE both guides the search space and performs tie-breaks when deciding membership to the archive. The algorithm's main loop is performed as follows:

The fitness of a solution candidate is determined by sum of the number of other solution candidates in both P and $\bar{P}$ that it strictly dominates. In scenarios where many solution candidates fail to strictly dominate other candidates a density estimation is introduced to prevent stagnation of selective pressures. The likelihood of this situation occurring increases with the number of fitness criteria because the odds of recombined solutions surpassing their competition along all axes becomes less probable. The density estimator utilized is the inverse Euclidean distance to the k^{th} nearest neighbour.

The truncation operator alluded to in step 5 is designed to maintain a constantly

Algorithm 3 Multi-objective GA: SPEA II

procedure PROCESSPOPULATION
$P \leftarrow$ Population of solution candidates
$\bar{P} \leftarrow$ Archive of solution candidates
$\bar{N} \leftarrow$ Maximum number of solutions in archive
top:
$P =$ **randomly initialized population of solution candidates**
$\bar{P} =$ **empty archive**
while $ExitCondition$ **is** $False$ **do**
evaluate solution candidates in P
$\bar{P} =$ **nondominated members of** P **and** $\bar{P}$
if number of solutions in $\bar{P} > \bar{N}$ **then**
reduce $\bar{P}$ **via truncation operator**
$P =$ **perform selection, recombination, and mutation on** $\bar{P}$

sized archive $\bar{P}$ in cases where the current number of nondominated solutions contained by $P + \bar{P} \neq \bar{N}$.

When the number of nondominated solutions is less than $\bar{N}$ it allows the most highly fit dominated solutions to enter the archive. When the number of nondominated solutions is greater than $\bar{N}$ the most crowded solution candidates are removed until the archive size equals $\bar{N}$

SPEA II more closely resembles the previously described genetic algorithms than PAES in that it includes a full population of solution candidates and applies selection, mutation, and recombination operators.

NSGA II

Among multi-objective GAs nondominated sorting genetic algorithm II, NSGA II[22], is considered state-of-the-art [78]. Several key advantages over other popular MOGAs include, reduced computational complexity, elitism, and parameterless fitness sharing operator. The algorithm can be outlined as follows.

Nondomination level is assigned such that no individuals on the same level strictly dominate any other individuals along all fitness axes. In other words, the first level consists of the population's best estimate of the Pareto set of solution candidates. Each subsequent level is the Pareto set of the population excluding all individuals contained by lower levels.

To maximize the distance between points on the current population Pareto front,

Algorithm 4 Multi-objective GA: NSGA II

procedure ProcessPopulation
$P \leftarrow$ Population of solution candidates
$Q \leftarrow$ Working population of solution candidates
$R \leftarrow$ Combination of P and Q together
$\bar{N} \leftarrow$ Maximum number of solutions in archive
top:
$P =$ **randomly initialized population of solution candidates**
while $ExitCondition$ **is** $False$ **do**
evaluate solution candidates in P
$Q =$ **perform selection, recombination, and mutation on** P
$R =$ **combine** P **and** Q
evaluate R **on nondomination level and partial rank**
$P =$ **most fit** $\bar{N}$ **solutions in** R

each level is internally sorted according the crowding distance between its nearest neighbours.

Even considering NSGA II's diminished computational complexity relative to it predecessor, NSGA [73], it remains expensive for large population sizes. As originally proposed NSGA II is also memory intensive. Namely, in order to reduce the time/computational complexity, storage requirements are on the order of N^2 which can limit population size in memory restricted environments.

2.5.4 Portfolio Selection

Given that the portfolio selection task can be viewed as a multi-objective optimisation problem it is not surprising that there are instances of genetic and evolutionary algorithm implementations. Due to the generalized nature of genetic algorithm's optimisation capability the areas of application vary significantly. Metaxiotis and Liagkouras' review of existing literature relating to multi-objective evolutionary algorithms for portfolio management [62] found that the vast majority of sampled research, $\approx 82.5\%$, contained 2 objectives. Moreover, the two most common objectives were asset's return means and variances. Despite similar objectives there were difference between these mean-variance portfolio selection problems when compared to mean-variance analysis [60]. The source of the difference frequently was founded in introduction of constraints.

The task of solving for the efficient frontier of portfolios in mean-variance analysis is typically achieved through quadratic or linear programming techniques.

Introducing constraints on objectives to improve realism or predictability can prove highly problematic for these methods. For instance, lack of a cardinality[4] constraints. When performing mean-variance portfolio optimisation the final efficient frontier can be improved by making as many assets available to the analysis as possible. An side-effect of the availability is the potential for a large number of those assets to contain a small allocation the portfolio's value. Many small positions can prove difficult to manage and may incur additional transaction costs when compared to portfolios with fewer, large positions. For these reasons an investor may want to limit the number of held assets. Use of multi-objective genetic algorithms to solve for the efficient frontier of portfolios with cardinality constraints was a natural application [3, 10, 17, 21, 56, 71]. Another approach to addressing the same concern is introduction of bounds on transaction lot sizes for assets [21, 53, 56]. Of sampled research in [62] relating to introduction of constraints over 88% addressed either cardinality or lot requirements. Other constraints include those on transaction costs [54] and total market sector allocation [72].

Among 2 objective works which did not concern constraints, some of the more unusual applications included revamping either risk or return objectives. In [19] the return objective function was found by combining multiple trading strategies' signals into portfolios with highest joint expected value. Examples of risk adjustment[7] involved introduction of uncertainty to the risk function[55] and substitution of the value-at-risk (VaR) metric[47].

Additional research pertains to including ternary or higher objectives. For example, Li and Xu simultaneously maximized a liquidity measure along with return mean and variance [52]. Doing so has the potential to reduce transaction costs and minimize slippage.

While genetic algorithms for portfolio optimisation have value in many instances there is significant potential for misuse. There are many circumstance in which GAs may by poorly suited to the optimisation task. Multiple continuous parameters is one of the cases where other machine learning methods could be considered. The reason genetic algorithms are the sole multi-objective optimisation method considered for the portfolio realignment task in later chapters is because the problem exists in discrete space. Discrete space optimisation is a field where genetic algorithms excel.

[4] a limit on the number of assets actively managed in a portfolio

2.6 Summary

Throughout the literature review and background chapter the topics most relevant to producing and evaluating the network model of financial markets were outlined. These subjects included:

1. Time Series Analysis
2. Algorithmic Trading
3. Artificial Markets
4. Modern Portfolio Theory
5. Genetic Algorithms

The time series analysis section introduced and demonstrated methods for modelling time series. This is relevant as network model is constructed from historic asset price time series. Residual tests, and boundaries of results were covered to allow estimation of model quality. The time series analysis section ended with the development of cointegration models. These models are a common basis for an equities trading strategy known as statistical arbitrage which led into the algorithmic trading section.

Statistical arbitrage was of particular significance in the algorithmic trading section as it bares some similarity to the the algorithmic trading validation implemented atop the proposed network model. The artificial market section discussed best practices and existing research in the field. Artificial markets were relevant to the thesis as they provide a transparent proving ground to evaluate the accuracy of the network model.

The modern portfolio theory section solidified the expected risk, expected return framework utilized when comparing investment opportunities. While MPT was not used for quantifying diversification benefits, as in its described purpose, it was used to recombine the sensor data which bases the network model. The section concludes with a discussion of widely accepted multi-objective genetic algorithms and their applications in portfolio management.

Information is the resolution of uncertainty.

- Cluade Shannon

3 The Network Model

Throughout this chapter the methodology applied to creating network model's of financial data is introduced and outlined. The intent is to unambiguously describe the process such that the work can be replicated at will. The logic, discussion, and rationale behind decisions made in each 'layer' of the algorithm are analyzed in future chapters. This chapter can be thought of as an introduction to the final process and a reference for implementation after considering later discussion.

3.1 Notation

The following table is a reference for notation. The listed variables are those most relevant to the network model or subsequent discussion. Any additional variables represent intermediate values lacking significance to the global scope. Those values are defined in place.

Variable	Description	Sample Value
N	Number of assets in data set	100
L	Number of observations/period per asset	252
C	Matrix of daily close prices	Matrix [N by L]
T	Index of target asset in C	Time-Series
X	Table of sensors	Table [w by N+1]
c	Constant term	7.84
ϵ	Error	Time-Series
s	Number of regressors per sensor	10
w	Number of sensors per target asset	5000
ψ	Weight applied to a regressor in a final model	0.11
ϕ	Weight applied to each regressor in a sensor	0.18
δ	Number of sensors per sensor portfolio	20
$\bar{z}$	Vector of sensor cumulative returns of length w	Vector [1 by w]
σ	Covariance matrix of sensor expected information content	Matrix [w by w]

3.2 Aims

The final aim of the proposed algorithm is to create a set of mean-reverting linear models which accurately predict long-run market trends. In a market with N assets a single example of a mean-reverting model, i.e. cointegration relationship, takes the form:

$$0 = \psi_1 C_1 + \psi_2 C_2 + ... + \psi_N C_N + c + \epsilon \tag{3.1}$$

Where:

$\psi_i =$ the scalar weight applied to the i^{th} regressor in a network relationship

$C_i =$ the i^{th} regressor or asset time series in a network relationship

Each mean-reverting linear model, hereafter referred to as a 'network relationship', is calibrated against a unique asset drawn from the pool of all assets. After the procedure is repeated for all assets there are a total number of network relationships equal to the number of assets, N. It is assumed that all network relationships represent a long-run relationship in the market. In the short-term it is possible for the relationships to break down but over time asset prices are expected to revert such that

all of the network relationships are satisfied. It is further assumed that the network relationships are unique.

If the network relationships calibrated on each asset are unique and represent a long-run relationship then error levels can be expected to indicate short-term market mispricings. When assets are mispriced it is possible to develop future price movement expectations. Furthermore, the weights denoted by ψ gave insight to linear associations among assets. One aim of the network model is to inspect these interactions to uncover inter-asset relationships which could not have been identified using traditional analysis.

Given that error in any long-run mean-reverting model indicates future price movements, the network model can be validated by observing the profitability of making trades based on these expectations. This use case is valuable because it is both an interesting application of market modelling and a validation of the models' out-of-sample accuracy. If simulated trades made based on the expectations of the network relationships prove profitable then the model is confirmed to be a meaningful representation of the market.

A further aim is to explore network relationships as a method for creating theoretically market neutral portfolios. In this context, market neutral portfolios are defined such that the long-run expectations for a given combination of assets are theoretically insulated from movements of the entire market. For example, trading the a portion/weight, ψ, of each asset, C, contained in any of the network relationships would create a portfolio of assets expected to long-run mean revert to a constant. This reversion to a constant is theoretically robust to overall market movements and is therefore considered market neutral. Note that weights ψ in any network relationship does not guarantee positive expected returns, only that long-run market neutrality is expected.

Continuing the aim to explore network relationships, the model is reviewed from the standpoint of systemic risk. The connectivity described by the network relationships can also be thought of as transmission system for financial contagion. These connections can be utilized to simulate shocks to multiple market sectors and observe aggregated exposures among sectors. This is an analysis available when using a network representation of markets and yields valuable information relating to market dynamics.

In summary, the ultimate aim of the thesis is to create a meaningful network model of financial markets that enables unique and valuable analyses. To this end, 3 tasks

are undertaken:

1. Create N unique linear models which mean-revert to 0 in the long-run
2. Asses the validity of the mean-reversion assumption by determining if asset prices move as expected based on current deviations
3. Assess the validity of network relationships by determining if a high level aggregation yields meaningful results

The latter 2 tasks have a double intent. While they are practical applications of the network models they are also validations. The nature of the network relationships makes it difficult to evaluate directly because there was no 'correct' answer for comparison. Thus, each network relationship was validated by reviewing its performance in quantitative applications. In short, verifying the accuracy of expectations drawn from the network models is possible; verifying the accuracy of the relationships themselves is not.

3.3 Method

In this section the procedure of developing a set of N network relationships from equity close prices is covered. The process is composed of 3 sub-processes dubbed 'layers'.

The technique begins with daily close prices of common stock for assets drawn from an index. The data were filtered to exclude assets which:

1. Contained fewer than 10 years of historic daily close data
2. Were no longer actively traded (survivorship bias)

The remaining stock price time-series were passed through to the 'Sensor Creation' layer i.e. Layer 1.

3.3.1 Sensor Creation: Layer 1

The sensor creation layer accepts stock price level data and 3 parameters as inputs. It outputs thousands of low quality 'sensors' containing small amounts of noisy information relating to market interactions. Each sensor is a linear model with s

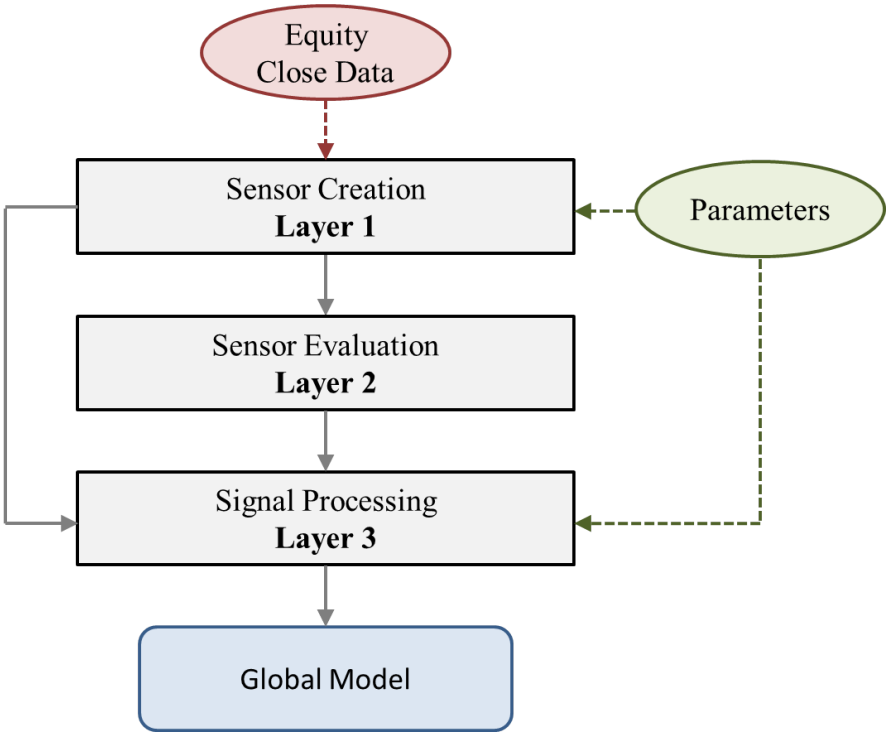


Figure 3.3.1: Network Model: Top Level Flowchart

randomly selected of asset return time-series regressed against a target asset's return series. The process is repeated w times yielding w sensors with randomly selected inputs for a common target asset. Pseudo-code can be found in algorithm 5.

The output of the procedure yields w sensors which are stored in a table where each row constitutes a single regression equation/sensor. The table contains $N+1$ columns to enable storage of all relevant sensor information i.e. weights and constants. The first N columns contain weights or ϕ for every possible regressor asset. This includes a value of $\phi = -1$ for the sensor's target asset, i.e. independent variable. The last value in the table stores the sensor's constant. The total number of non-zero entries in each row of the table is $s+2$ representing the s weights or ϕ values, a $\phi_T = -1$ in the target asset's column and a constant in the last column. Consider the example below where table 3.3.1 is a sensor or row in the output, X, and table 3.3.2 is an excerpt from X for target asset 2. Relevant parameters for the example are: $[s, N, T] = [4, 10, 2]$

Algorithm 5 Network Model: Sensor Creation

procedure CREATESENSORS
  $s \leftarrow$ Number of regressors per sensor
  $w \leftarrow$ Number of sensors per target asset
  $N \leftarrow$ Number of assets in data set
  $L \leftarrow$ Number of observations/periods per asset
  $C \leftarrow$ Matrix of Daily Close Prices [N by L]
  $T \leftarrow$ Index of target asset in C
  $X \leftarrow$ Table of sensors
top:
  $T = 2$
  for $j = 1 : w$ **do**
    $AvailRegressors = 1 : N$
    remove T **from** $AvailRegressors$
    $Regressors =$ **randomly select** s **values from** $AvailRegressors$
    $TargetSeries = C(T, :)$
    $RegressorSeries = C(Regressors, :)$
    $RegressorSeries =$ **append vector of** 1 **of length** L **to** $RegressorSeries$
    $SensorWeights =$ **regress** $RegressorSeries$ **on** $TargetSeries$
    append $SensorWeights$ **to** X

Table 3.3.1 Network Model: Single Sensor

ϕ_1	ϕ_2	ϕ_3	ϕ_4	ϕ_5	ϕ_6	ϕ_7	ϕ_8	ϕ_9	ϕ_{10}	c
0.24	-1.00	0.00	0.00	0.00	-0.04	0.30	0.00	0.00	0.67	41.05

The comparable equation to table 3.3.1 is:

$$0 = 0.24C_1 - 1.00C_2 - 0.04C_6 + 0.30C_7 + 0.67C_{10} + 41.05 + \epsilon \tag{3.2}$$

When each sensor in the format of 3.3.1 is added to a table, the end result for the second target asset is similar to table 3.3.2.

Table 3.3.2 Network Model: Multiple Sensors

ϕ_1	ϕ_2	ϕ_3	ϕ_4	ϕ_5	ϕ_6	ϕ_7	ϕ_8	ϕ_9	ϕ_{10}	c
0.24	-1.00	0.00	0.00	0.00	-0.04	0.30	0.00	0.00	0.67	41.05
0.00	-1.00	0.00	0.35	0.00	0.00	0.46	-0.18	0.00	0.11	11.77
0.00	-1.00	0.00	0.28	-0.23	-0.31	0.00	0.00	0.00	0.00	61.97
0.00	-1.00	0.00	0.00	0.00	-0.31	0.25	0.00	0.29	-0.24	50.30
0.05	-1.00	0.00	0.00	0.00	-0.18	0.00	-0.19	0.13	0.00	58.78

There are 2 primary advantages to storing results in this fashion. First, the output is human readable. Second, it allows for straightforward reconstruction of the sensors' in-sample error. The reconstruction is possible by multiplying the close price matrix, C, with the first N elements of any row vector or sensor in the sensor table and adding its final element to the result. If the randomly selected row is i:

Algorithm 6 Network Model: Sensor Approximate Reconstruction

procedure SensorError
  $N \leftarrow$ Number of assets in data set
  $C \leftarrow$ Matrix of Daily Close Prices [N by L]
  $X \leftarrow$ Table of sensors
  $\epsilon \leftarrow$ Error time-series in sensor
  $w \leftarrow$ Number of sensors per target asset
  $i \leftarrow$ Random positive integer $< w$
top:
  $\epsilon = X(i, 1 : N) * C + X(i, N + 1)$

A concise version of the sensor creation and error reconstruction processes have been implemented in cross compatible Octave and MATLAB code and can be seen in figures 3.3.2 and 3.3.3 respectively.

```
%% Create Sensors
% Set Workspace/Parameters
s = 6; % Number of assets per sensor
w = 50; % Number of sensors per target asset
N = 20; % Number of assets in data set
L = 250; % Observations per Artificial Financial Time Series
Factors = cumsum(rand([N,L])-.5,2)+50; % Artificial Market Factors
for i = 1:N % Artificial Assets Based On Factors
    C(i,:) = (rand(1,8)/4)*Factors(randsample(1:N,8),:);
end
T = 1; % Target Asset index in C
% Perform Regressions
X = zeros([w,N+1]);
for j = 1:w
    allregressors = 1:N;
    allregressors(T) = [];
    regressors = randsample(allregressors,s);
    targetseries = C(T,:);
    regressorseries = vertcat(C(regressors,:),ones([1,L]));
    X(j,horzcat(regressors,N+1)) =...
        regress(targetseries',regressorseries');
    X(j,T) = -1;
end
clear j idx targetseries regressors allregressors regressorseries
```

Figure 3.3.2: Network Model: Sensor Creation MATLAB Code

The reconstruction and plotting of a single sensor's approximation of its target asset can be performed using the following code.

```
%% Reconstruct Example Sensor
i = 4; % Randomly selected sensor
targetseries = C(T,:);
epsilon = X(i,1:N)*C+X(i,N+1); % Sensor residual series
```

Figure 3.3.3: Network Model: Sensor Approximation of Target Asset MATLAB Code

Plots of the error series and approximation of the target asset, T, are available in

figure 3.3.4.

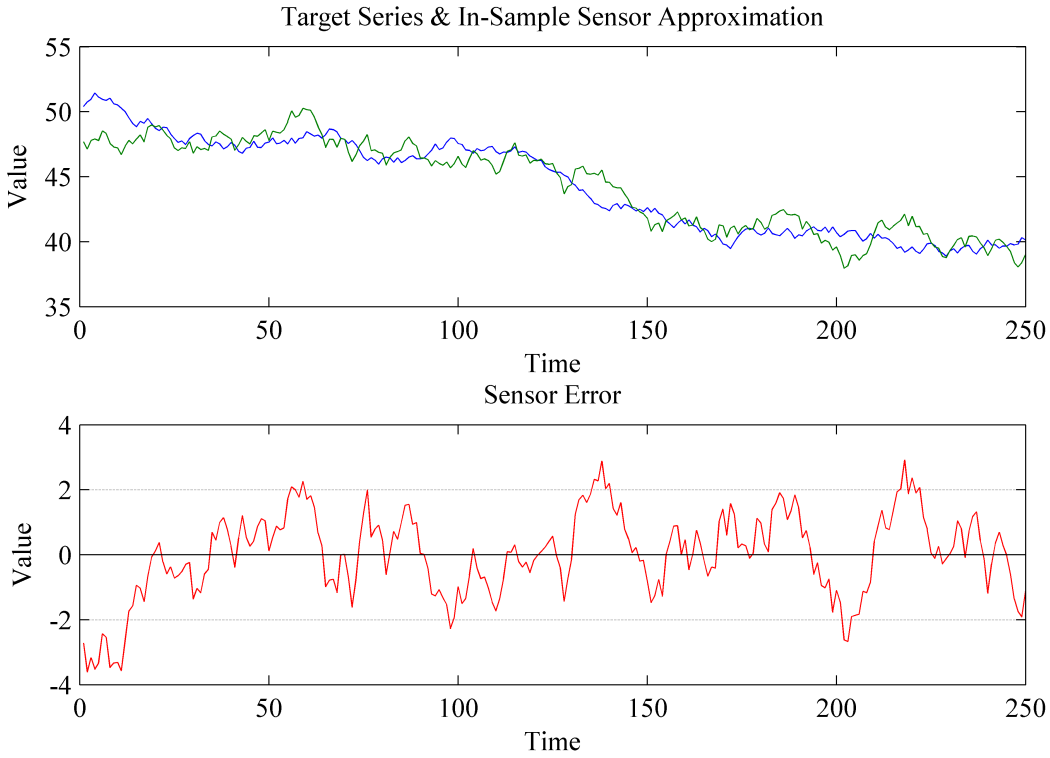


Figure 3.3.4: Network Model: Sensor Approximation Graphs

The sensors produced throughout the examples in code are based on randomly produced autoregressive series which take the place of market factors. The assets are linearly weighted combinations of randomly selected subsets of the factors. The code is intended the demonstrate correct computations and enable small scale experimentation. Discussion in later chapters explains why the proposed approach uncovers shared information through the sensor creation process in financial markets.

When applied to true market data the created sensors independently contain a small amount of information about the target asset largely corrupted by idiosyncratic noise. In order to further the analysis towards the aim of a single meaningful network relationship per target asset, the sensors are processed into a smaller number of higher quality sensor portfolios. To that end, the table of sensors, X, found in layer 1 is passed to the layer 2 for evaluation.

3.3.2 Sensor Evaluation: Layer 2

To understand the need for a sensor evaluation layer it is necessary to understand the goal of the signal processing layer, or layer 3. The premise of the signal processing layer is to combine rows of the sensor table from layer 1 to improve sensor quality beyond a simple average and ensure that the information contained in the output reflects the proportions of the underlying target asset. To achieve this, the quality and uniqueness of sensors must be evaluated. It is assumed that when sensors are high in quality, taking trading positions based on their information content generates positive returns. A higher quality sensor has more consistently profitable, i.e. lower variance, returns. Therefore, the quality of each sensor is computed in terms of in-sample prediction accuracy, or returns, and return variance. Both metrics are evaluated in a simulated trading algorithm. The expected return and expected risk during simulated algorithmic trading is the basis for outputs in the sensor evaluation layer.

Expected positive return trading positions based on sensor expectations are possible due to the expectation of long-run mean-reverting tendencies in the sensor models. When error in a sensor's approximation of its target asset is large the target asset's price is expected to change such that error returns to 0.00. The accuracy and consistency of each sensor's predictions are compared in terms of expected risk and expected return by simulating appropriate positions based on the sensor's error term. All values for this layer remain in-sample.

In summary, the quality of each sensor was determined by computing its in-sample expected risk and expected return in a simulated algorithmic trading strategy. The trading strategy implemented consists of trading only the target asset's side of the spread according to the pairs trading strategy as described in chapter 2. The process is seen in algorithm 7.

Algorithm 7 Network Model: Sensor Evaluation

procedure EVALUATESENSORS

$w \leftarrow$ Number of sensors per target asset

$N \leftarrow$ Number of assets in data set

$L \leftarrow$ Number of observations/period per asset

$C \leftarrow$ Daily close data matrix [N by L]

$T \leftarrow$ Index of target asset

$X \leftarrow$ Table of sensors

$R \leftarrow$ Matrix of in-sample sensor returns series [w by L-1]

$\hat{\zeta} \leftarrow$ Vector of normalized sensor signals/residuals

$\bar{z} \leftarrow$ Vector of sensor cumulative returns of length w

$\sigma \leftarrow$ Covariance matrix of sensor expected information content [w by w]

top:

$T = 1$

for $k = 1 : w$ **do**

$sensor = X(k, :)$

$targetseries = C(T, :)$

$\zeta = sensor(1 : N) * C + sensor(N + 1)$

$\hat{\zeta} = -\zeta / \mathbf{max}(|\zeta|) \leftarrow$ Residuals normalized to bound -1 to 1

$Tret = 1$ **period difference of** $targetseries$

$r = (1$ **period difference of** $\hat{\zeta}) * Tret \leftarrow$ Daily return from trading

$\bar{z}(k) = \mathbf{sum}(r) \leftarrow$ Cumulative return from frictionless trading

$R(k, :) = r$

for $i = 1 : N$ **do**

$\sigma =$ **Covariances among rows in** $R \leftarrow$ Covariance of sensor returns

Normalizing ζ and then multiplying by -1 returns an in-sample trading signal bound between -1 and 1. The trading signals are the residuals of the sensor divided by the largest magnitude it contains. Multiplying the changes, i.e. 1 period difference, of the trading signal by the returns of the target asset computes the frictionless return associated with a holding $\hat{\zeta}$ portion of the target asset at each time period. When spreads are large, the investment in the target asset is large and conversely small spreads indicate small investments. Returns are only realized on the portion of investment which changes between periods.

The metrics $\bar{z}$ and σ corresponding with return and covariances for every sensor in table of sensors for a single target asset. The mutually identified information among sensors is determined by the covariance matrix of sensor returns with a common target asset. Computation of these metrics for a single target asset in Octave/MATLAB code is shown in figure 3.3.5.

```
%% Evaluate Sensors
R = zeros([w,L-1]);
zbar = zeros([w,1]);
for k = 1:w; % For each sensor
    targetseries = C(T,:);
    zeta = X(k,1:N)*C+X(k,N+1);
    % Compute Returns and Risk From In-Sample Trading
    nzeta = -zeta/max(abs(zeta));
    rTar = diff(targetseries);
    r = diff(nzeta).*rTar;
    R(k,:) = r;
    zbar(k) = r(end); % Trading returns
end
Sigma = cov(R'); % Trading covariances
clear k R r rTar nzeta targetseries zeta
```

Figure 3.3.5: Network Model: Sensor Evaluation Risk & Return MATLAB Code

The evaluation of sensors along 2 common metrics provides the information necessary construct higher quality sensors along the same criteria. To make these improvements, information from the low quality 'Sensor Creation' layer and the 'Sensor Evaluation' layer are needed. The improvement process is performed by Layer 3 or the 'Signal Processing' layer.

3.3.3 Sensor Processing: Layer 3

The sensor processing layer aims to convert a large number of sensors from layer 1 into a reduced number of high quality sensor portfolios. Quality, as discussed in the sensor evaluation layer, is embodied by an increase in information content and signal to noise ratio. These features are measured by computing the expected risk and covariance via simulated algorithmic trading with each sensor. With those values as inputs, the 'Sensor Processing' layer is able to create a reduced number of higher quality sensor portfolios.

Bootstrap aggregating, i.e. averaging, gives equal value to all available models in the collection. From pesudo-code, the ensemble of available models is X. Each model in the ensemble, i.e. row in X, is given equal consideration or vote in contributing to the ensemble's expected response. The process in pseudo-code is presented in algorithm 8

Algorithm 8 Network Model: Bootstrap Sensor Processing

procedure Bootstrap Aggregating
    $X \leftarrow$ Table of sensors
    $w \leftarrow$ Number of sensors per target asset
top:
    $NetworkRelationship = (\textbf{sum columns of } X)/w$

Despite its simplicity the approach does generate meaningful out-of-sample models. The equivalent Octave/MATLAB code in figure 3.3.6. This provides the benchmark to exceed when developing alternate techniques.

```
%% Signal Processing: Bootstrap Aggregating
NetworkRelationship = mean(X,1);
```

Figure 3.3.6: Network Model: Bootstrap Aggregating MATLAB Code

Returning to the domain specific signal processing technique, the designation of higher quality sensors as 'portfolios' was not arbitrary. Each sensor with it's risk and return values can be treated like an asset in a portfolio. When creating higher quality sensor portfolios we aim to maximize the return of the portfolio while minimizing the risk. Moreover, the measure of risk allows for diversification by minimizing the

covariance among asset returns in any sensor portfolio. An output of this process is the mean-variance optimal efficient portfolio discussed in chapter 2. Algorithm 9 shows the process for accepting sensors and sensor evaluation data and returning a network relationship.

Algorithm 9 Network Model: Signal Processing

procedure SIGNALPROCESSING
$N \leftarrow$ number of assets in data set
$\delta \leftarrow$ number of sensors per sensor portfolio
$\sigma \leftarrow$ covariance of returns for sensors $[w$ by $w]$
$\bar{z} \leftarrow$ returns for sensors $[1$ by $w]$
$w \leftarrow$ Number of sensors per target asset
$psensors \leftarrow$ Table containing sensor portfolios $[w/\delta$ by $N+1]$
top:
$\delta = 20$
for $i = 1 : w/\delta$ **do**
$idx = ((i-1)\delta) + 1 : i * \delta$
$SubSigma = \sigma(idx, idx) \leftarrow$ Excerpt of σ for sensors in the current portfolio
$SubZbar = \bar{z}(idx) \leftarrow$ Excerpt of $\bar{z}$ for sensors in the current portfolio
$uvec =$ **vector of** 1 **of size** δ
$OptWts = (SubSigma^{-1} * SubZbar)/(uvec' * SubSigma^{-1} * SubZbar)$
$psensors(i,:) = OptWts' * X(idx,:)$
$NetworkRelationship = ($**sum columns of** $X)/(w/\delta)$

The algorithm splits the sensors into groups of size δ. These sensors are treated as assets in a portfolio. Each set of sensors yield an efficient frontier of recombination weights. The selected weights for sensor recombination is the set represented by the point of tangency of a line from the origin to the efficient frontier. This point also represents the portfolio with the highest Sharpe ratio.

```
%% Signal Processing
sPp= 5; % sensors per sensor portfolio
psensors = zeros([floor(w/sPp),N+1]);
for i = 1:floor(w/sPp)
    idx = ((i-1)*sPp)+1:i*sPp;
    OptWts = (Sigma(idx,idx)^-1*zbar(idx))/...
        (ones([sPp,1])'*Sigma(idx,idx)^-1*zbar(idx));
    psensors(i,:) = OptWts'*X(((i-1)*sPp)+1:i*sPp,:);
end
clear i idx OptWts
% Mean of portfolio sensors creating a network relationship
NetworkRelationship = mean(psensors,1);
```

Figure 3.3.7: Network Model: Signal Processing MATLAB Code

The aggregation of portfolio sensors utilizes bootstrap aggregating as previously defined. The values in each sensor portfolio created for a single target asset are averaged creating 1 network relationship. Every network relationship represents the final output from the entire process. The set of all network relationships, 1 per asset, defines a completed network model.

3.4 Summary

A process to create N network relationships of a stock market was outlined. The process began with common stock close prices for a large set of assets. The first step was to create a thousands of sensors for all assets. Each sensor consisted of a linear model comparing a single target asset and a small set, s, of randomly selected regressor assets. The sensors were expected to reveal small amounts of information about their target asset.

Once a sufficient number of sensors, set by parameter w, were created their quality was evaluated. Quality was determined according to how well expectations extracted from each sensor generated profit in a simulated algorithmic trading strategy. The 2 measures of quality considered were total in-sample returns and in-sample covariances.

With 2 measures of quality for every sensor, the w sensors were divided into equal groups of size δ. The groups, with associated covariances and returns, were taken to

be assets in a portfolio. The Pareto front of trade-offs between the 2 metrics was found for each group, i.e. efficient frontier. After selecting the optimal point on the Pareto frontier the sensors were linearly weighted according to the point's underlying asset/sensor allocations. These sensor portfolios represented higher quality linear models of a single target asset. The final step in creation of a network model for the target asset was to apply bootstrap aggregating to portfolio sensors. This process resulted in 1 network relationship for each target asset.

The method is repeated with each asset selected to be the target asset creating N network relationships. Each of these models is expected to represent a long-term trend which mean-reverts to a constant value of 0.

After the network models are created for all target assets, the next challenge is measuring their validity. As there is no market in which dependencies are entirely known, nor asset price construction fully understood, the accuracy of any long-run relationships can not be directly assessed. Furthermore the iterative and incremental process used in the network relationships' creation does not lend itself to establishing statistical significance. Instead, the value of network models is evaluated through multiple applications.

Computer science is no more about computers than astronomy is about telescopes.

- Edsger Dijkstra

4

Sensor Creation: Layer 1

4.1 Aims

The sensor creation layer aims to construct a collection of weak learners containing small amounts of information relating to the interactions of a single selected asset with the remainder of the market. The method utilized to create these weak learners is outlined in the next section. The chapter aims to:

1. Create a collection of weak learners
2. Test the learners for valuable information content in an artificial market

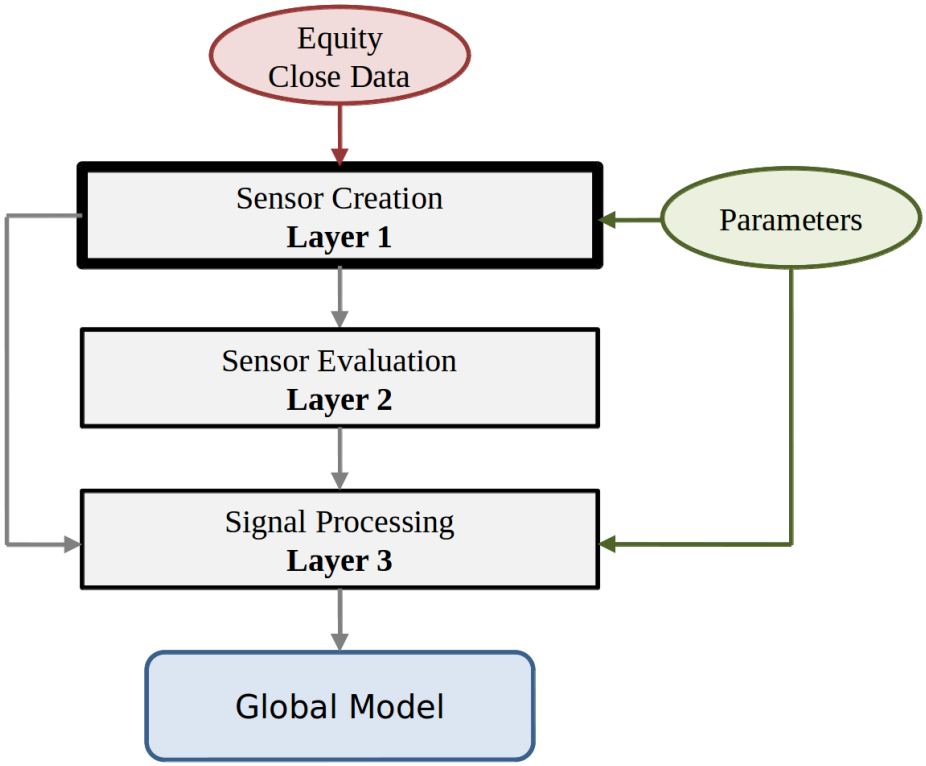


Figure 4.1.1: Sensor Creation: Level Flowchart

4.2 Method

The sensor creation layer initializes a large number of models of a single target asset selected from the market. These models are created by attempting to reconstruct the target asset using a small subset of other assets. Performing the process thousands of times with a unique subsets of regressor assets yields thousands of low quality models. The approach selected for reconstructing target assets was multiple regression.

Given that each model is low in statistical quality they are renamed 'sensors'. The distinction is made to illustrate that no decisions should be made on the sensors in this basic form. Despite the sensors' low quality, it is expected for each sensor to contain a small amount of information about the underlying target asset. Furthermore, this information is expected to remain accurate out-of-sample.

Each sensor, i.e. linear model, is expected to contain information relating to the target asset's movements. This expectation is made possible due to assumptions relating to the pricing of assets. It is assumed:

1. Market externalities, i.e. factors, impact asset prices
2. Some assets share dependencies on market factors
3. Assets in a common market sectors are more likely to share dependencies on factors
4. Asset external dependencies change slowly

These assumptions are critical to the underpinnings of the proposed process. Market externalities are hereafter referred to as 'factors'. Assuming that factors impact asset prices and multiple assets can share factor dependencies creates possibilities for modeling inter-asset relationships. Specifically, it enables approximating 1 asset using a combination of other assets. If a linear combination of assets' factor dependencies closely approximate a target asset's dependencies then price movements attributable to their factor dependencies are also expected to be approximately equal. Furthermore, if asset dependencies change slowly the discovered relationship should hold out-of-sample. Independently, the sensors do not contain sufficient information to generate meaningful out-of-sample value. However, a large number of sensors combined into a single model may contain sufficient information to create out-of-sample value.

Information can be extracted from sensors through knowledge of the linear relationships and regressor assets contained within. It is not an aim of the sensor creation layer to leverage this information into a useful output. The sensor creation process intends to explore the dataset. It is initially validated by creating an average model and applying it in an artificial market.

The sensor creation layer is responsible for 'extracting' noisy associations among subsets of assets in the market. The remainder of the network model creation process is designed around compiling these associations into a meaningful model.

4.2.1 Notation

The table below defines all variables and parameters used throughout the sensor creation layer. The notation was kept identical to previous descriptions in pseudo code. The relevant values from chapter 3 with new additions for the artificial market are listed below.

Variable	Description	Sample Value
N	Number of assets in data set	100
C	Matrix of daily close prices	Matrix [N by L]
L	Number of observations/period per asset	252
T	Index of target asset C	Time-Series
X	Table of sensors	Table [w by N+1]
c	Constant term	7.84
s	Number of regressors per sensor	10
w	Number of sensors per target asset	5000
ϕ	Weight applied to each regressor in a sensor	0.18
z	Number of factors per asset	15
l	Length of training data	252
n	Number of artificial factors	70

4.2.2 Process

Pseudo code identical to the exert from Chapter 3 can be found in algorithm 10.

Algorithm 10 Network Model: Sensor Creation

procedure CreateSensors
  $s \leftarrow$ Number of regressors per sensor
  $w \leftarrow$ Number of sensors per target asset
  $N \leftarrow$ Number of assets in data set
  $L \leftarrow$ Number of observations/periods per asset
  $C \leftarrow$ Matrix of Daily Close Prices [N by L]
  $T \leftarrow$ Index of target asset in C
  $X \leftarrow$ Table of sensors
top:
  $T = 2$
  for $j = 1 : w$ **do**
    $AvailRegressors = 1 : N$
    remove T **from** $AvailRegressors$
    $Regressors =$ **randomly select** s **values from** $AvailRegressors$
    $TargetSeries = C(T, :)$
    $RegressorSeries = C(Regressors, :)$
    $RegressorSeries =$ **append vector of** 1 **of length** L **to** $RegressorSeries$
    $SensorWeights =$ **regress** $RegressorSeries$ **on** $TargetSeries$
    append $SensorWeights$ **to** X

The sensor creation layer begins with a matrix of common stocks' close levels. Each asset is select in turn to be the 'target asset' or 'target'. All sensors for a selected target are created before the target is iterated to the next asset. Every sensor is a linear model equating the target time-series with a fixed number, s, of regressor assets and a constant.

Each of the assets time-series selected to be one of the s regressors is randomly selected, without replacement, from the pool of all N assets excluding the current target. The regressors and a column vector of ones to allow for a constant term are entered into the multiple regression equation.

The output sensors are stored in a table, X, where each row represents a sensor. This output takes the form of table 4.2.1.

Table 4.2.1 Sensor Creation: Multiple Sensors

ϕ_1	ϕ_2	ϕ_3	ϕ_4	ϕ_5	ϕ_6	ϕ_7	ϕ_8	ϕ_9	ϕ_{10}	c
0.24	-1.00	0.00	0.00	0.00	-0.04	0.30	0.00	0.00	0.67	41.05
0.00	-1.00	0.00	0.35	0.00	0.00	0.46	-0.18	0.00	0.11	11.77
0.00	-1.00	0.00	0.28	-0.23	-0.31	0.00	0.00	0.00	0.00	61.97
0.00	-1.00	0.00	0.00	0.00	-0.31	0.25	0.00	0.29	-0.24	50.30
0.05	-1.00	0.00	0.00	0.00	-0.18	0.00	-0.19	0.13	0.00	58.78

The table contains a column for each asset in the dataset along with one extra column for a constant term. Every row represents a single sensor. Each ϕ corresponds to the regressed weight for the randomly selected subset of potential regressors. For example, the first row of table 4.2.1 would be represented by the following equation:

$$0 = 0.24C_1 - 1.00C_2 - 0.04C_6 + 0.30C_7 + 0.67C_{10} + 41.05 + \epsilon \tag{4.1}$$

Where:

$C_i =$ the i^{th} regressor or asset time series in a network relationship

4.2.3 Discussion

The reason each sensor is expected to yield information about market associations relates to the assumptions made at the beginning of the chapter. We assumed that markets are at least partially priced through changes in external factors. Moreover, dependencies on some of these factors are shared among assets. Resultantly, it may be

possible to approximate a single asset, i.e. the target asset, through a combination of other assets, i.e. the regressors.

The random sampling of assets to be used as regressors applies similar logic. Since we assume assets are dependent on external factors, every unique combination of regressors presents a new combination of external factor dependencies. The ability to determine a target asset's dependencies on factors is limited. The problem would be much simpler if the factors could be observed directly. Instead, we attempt to approximate the target asset's dependencies by iteratively creating weighted combinations other assets. In the long-term the aim is to create a single linear model for each target asset composed of all other assets in the market. If the final combination of other market assets sufficiently approximates the external dependencies of the target asset then the model should hold out-of-sample.

Residual diagnostics for sensors need not be as robust as typical regression. In particular it is not necessary to ensure that residuals are not autocorrelated. The relationships being sought are long-run in nature. For residuals to lack autocorrelation the market would need to mean-revert to sensors' expectations more frequently than every day. Given each cointegration equation's use as a weak learner it was determined that removal of poorly specified models was not necessary. Since these sensors are intended as weak learners, no statistical testing of residual values or cointegrating equation significance took place. Furthermore, filtering the sensors may limit the model's ability to detect available information from weakly relevant assets.

In summary, the task is to approximate 1 unknown target asset with a large selection of other unknowns. The sensor creation layer provides a method of increasing the number of scenarios available for a user to probe the target asset. Due to the opaque nature of financial markets it proves difficult to validate the intermediary step quantitatively. To address this issue, an artificial market was created.

4.3 Validations

The ability to validate of the sensor creation layer is limited for 2 reasons. First, the model is incomplete. Testing an intermediary step does not allow validation of the final 'output' or network relationship. Second, there is no correct solution to compare an output to.

Testing of an intermediary step is made possible averaging all sensors for a target

asset to create an temporary, and noisy, network relationship to work from. While this approach does not create network relationships as robust as the proposed method it is sufficient to show predictability in certain circumstances.

The absence of a correct solution for comparison eliminates the option of direct assessment of quality. To compensate, the sensors are evaluated by generating expectations in different applications. The output from each application can be compared to expected output allowing for indirect validation of the sensors' information contents.

4.3.1 Artificial Market

Framework

The purpose of the artificial market validation is to demonstrate that the sensor creation layer extracts meaningful information relating to market dynamics when the assumptions made about asset pricing are known to be correct.

The artificial market was designed as a collection of assets generated from a pool of random market factors. Each asset was dependent on a small subset of factors. This created an environment where crucial assumptions made about asset pricing were guaranteed and quantifiable. Those assumptions were:

1. Market externalities, i.e. factors, impact asset prices
2. Some assets share dependencies on market factors
3. Assets in a common market sectors are more likely to share dependencies on factors
4. Asset external dependencies change slowly

In this case, asset external dependencies do not change. Once an asset's dependencies were set, they remained constant. The artificial market did not contain any clustered market sectors.

A total of 70 factors were created. These factors were $AR(1)$ series as in equation 2.13. Each asset in the artificial market was defined as a linear combination of 15 randomly selected factors with constants and Gaussian noise. The weight vector applied to factors consisted of positive random weights with unit length. Constant

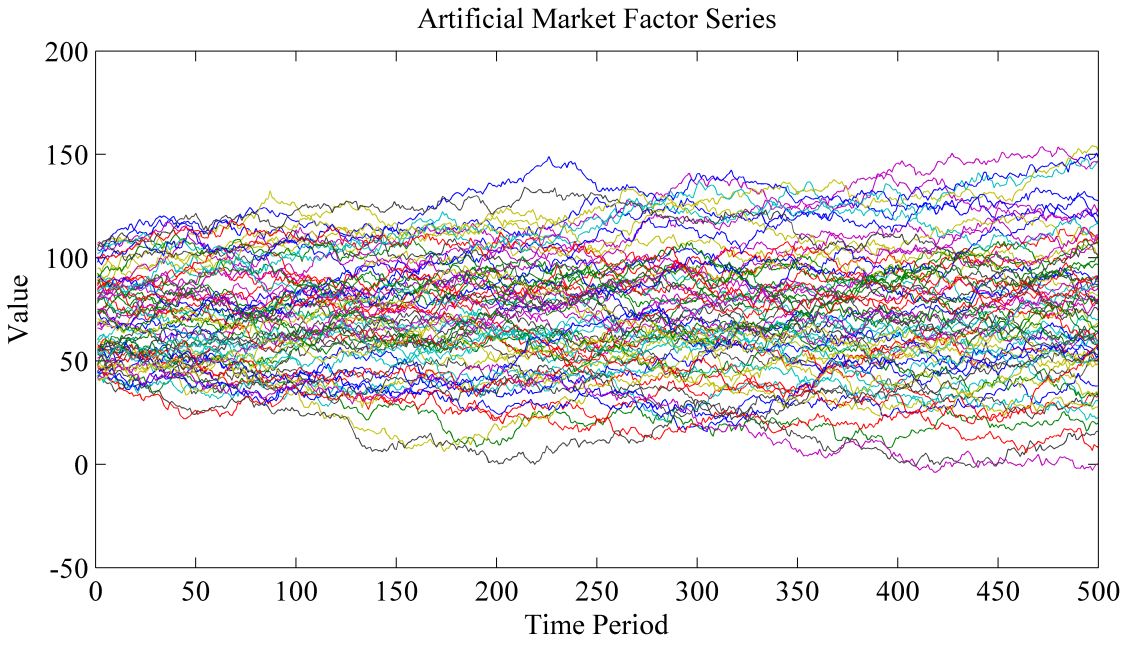


Figure 4.3.1: Artificial Market: Factors

terms were uniformly random between 45 and 105. The factors are visible in figure 4.3.1.

$1,000$ assets were constructed from the 70 factors. Assets were composed of 15 randomly selected factors to create a diverse and economy where all assets are not positively correlated. Those factors were assigned a normalized random vector of positive weights. Gaussian noise was introduced with mean of 0 and variance of 2.5. Pseudo-code applied to create artificial assets can be seen in algorithm 11.

Estimating Factor Composition

With the artificial market created, the approach outlined in algorithm 10 could be applied. The benefit to testing the algorithm in an artificial market is that the factor composition of all assets is known. This means it is possible to determine if the factor composition of the asset being modelled, and the model align. if the model is expected to generate any out-of-sample predictability this is a requirement. If the factor composition of the model and the target asset do not align then the algorithm was not capable of identifying meaningful structure among assets. The relevant parameters to

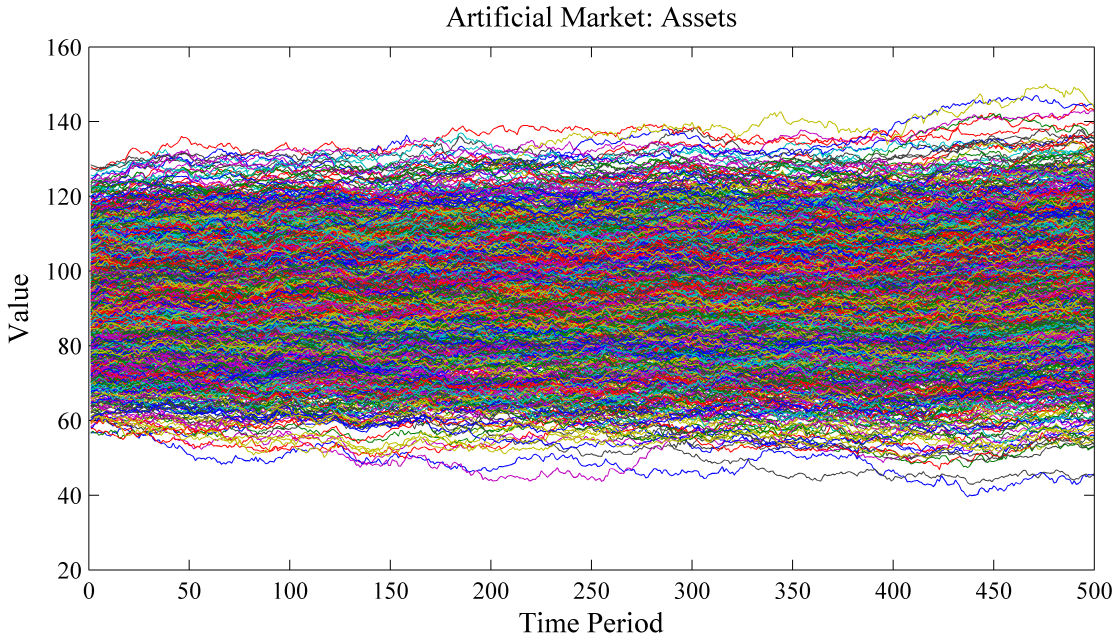


Figure 4.3.2: Artificial Market: Assets

Algorithm 11 Artificial Market: Asset Creation

procedure CREATEARTIFICIALASSET
$N \leftarrow$ Number of assets in data set
$n \leftarrow$ Number of factors
$L \leftarrow$ Number of observations/periods per asset
$C \leftarrow$ Matrix of Simulated Close Prices [N by L]
$F \leftarrow$ Matrix of factor values [n by L]
$z \leftarrow$ Number of factors per asset
top:
$N = 1000$
$L = 500$
$z = 15$
for $k = 1 : N$ **do**
$AvailFactors = 1 : 70$
$SelectedFactorIndex =$ **randomly select** z **values from** $AvailFactors$
$SelectedFactorSeries = F(SelectedFactorIndex, :)$
$uv =$ **uniformly random values of between** 0 **and** 100 **of length** z
$\hat{uv} = uv/|uv| \leftarrow$ Normalize weights
$Constant =$ **randomly select values between** 45 **and** 105
$Noise =$ **values from Guassian distribution with mean** 0 **and variance** 2.5
$ArtificialAsset = \hat{uv} * SelectedFactorSeries + Constant + Noise$
append $ArtificialAsset$ **to** C

Table 4.3.1 Parameters required for creating sensors in the artificial market

Parameter	Value
s	8
w	5000
l	252

execute algorithm 10 on the artificial market data are listed in table 4.3.1

As the process was applied purely for demonstrative purposes, asset 1 was arbitrarily chosen as the target asset for modelling. In order to construct a single model from the 5000 sensors the bootstrap aggregating approach outlined in chapter 3 was used. The process is reiterated in algorithm 12.

Algorithm 12 Sensor Creation: Bootstrap Sensor Processing

procedure BOOTSTRAP AGGREGATING
$X \leftarrow$ Table of sensors
$w \leftarrow$ Number of sensors per target asset
top:
$NetworkRelationship = (\textbf{sum columns of } X)/w$

After realizing a single network relationship for asset 1 the factor dependencies of the asset and the model's estimation of the asset or 'aggregate asset' could be compared. The best estimate of asset 1's factors is found in figure 4.3.3.

It was found that the Aggregate asset's composition was similar to asset 1's composition with a cosine angle similarity metric between asset 1's composition and the aggregate of 0.863. This value becomes the benchmark to exceed in later layers of the network model when applying sensor processing techniques.

Since cointegrating network relationship is robust to linear changes in levels, constant terms could be ignored. For the same reason, it was acceptable to divide the result through by a constant,i.e. linearly scale, to create similarity in levels between target asset 1 and the simulated 'aggregate asset'. The plot of asset 1 and the aggregate asset offset by 15 value units is presented in figure 4.3.4.

The curves visually track which further suggests that the model's estimation is valuable. Note that values after period 250 were out-of-sample. To assess a possible application of the model, asset 1 and its aggregated sensor counterpart could be used in the simple pairs trading strategy outlined in chapter 2.

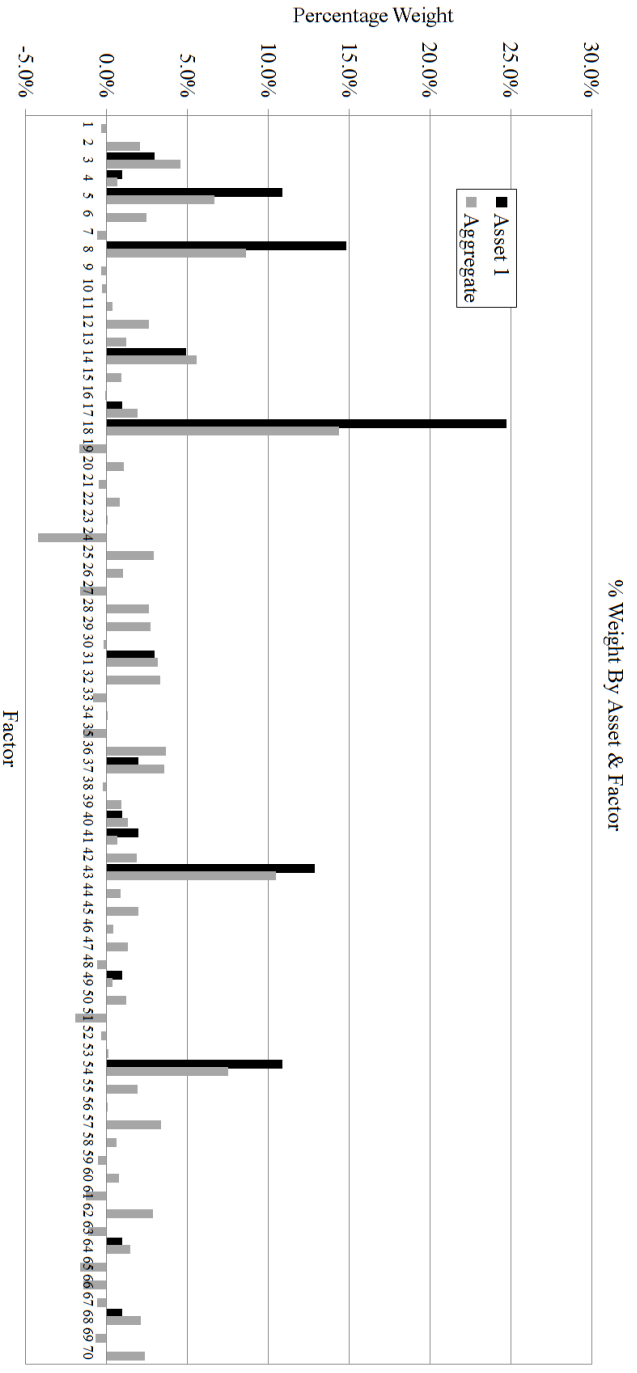


Figure 4.3.3: Artificial Market: % Weight by Asset and Factor via Sampling

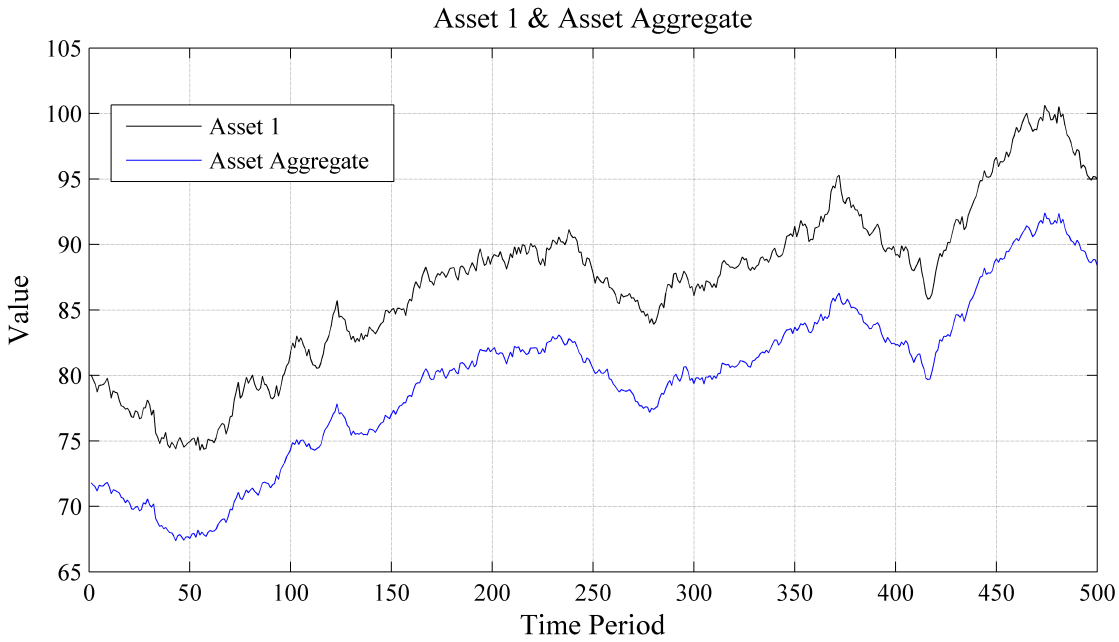


Figure 4.3.4: Artificial Market: Close Price Asset 1 and Aggregate Asset

Pairs Trading Application

Thus far, the emphasis in the sensor creation layer was applied to the problem of estimating a vector of known factors contributing to each artificially constructed asset. While useful for validating the model under controlled circumstances it is also possible to validate that the model has identified meaningful structure when underlying factors composing assets are not known.

As in the background chapter, a pairs trading strategy was applied. If a trading strategy yielded out-of-sample profits then the expectation that the network relationship is a meaningful representation of long-run market dynamics is supported. Figure 4.3.5 is the equivalent output to figure 2.2.4.

Plotting the out-of-sample profitability of the simple pairs trading strategy where profits/losses for the aggregate and asset 1 are tracked separately and jointly returns figure 4.3.6.

The pairs trading strategy shows consistent return from asset 1 with a Sharpe ratio of 2.4167 and consistent losses for the aggregate asset with a Sharpe of -0.9532. The more relevant metric is the portfolio of both aggregate and asset 1. The ratio for this combination was 3.6684. Despite the negative returns of the aggregate asset, these values sufficiently hedged asset 1's returns such that Sharpe ratio of the portfolio was

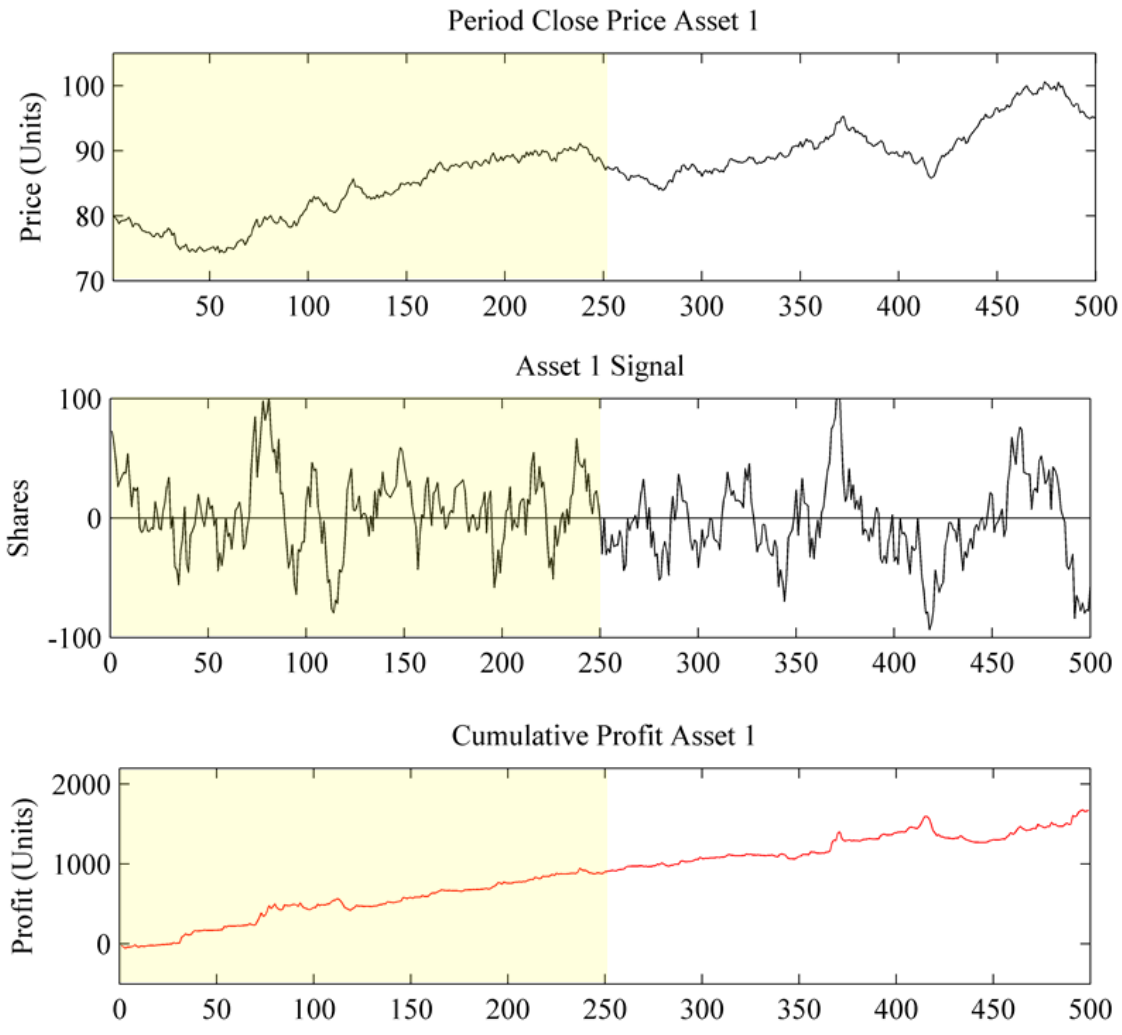


Figure 4.3.5: Artificial Market: Asset 1 Close Price, Signal and Profit

largest. This behaviour was in line with expectations for a valid pairs trading set.

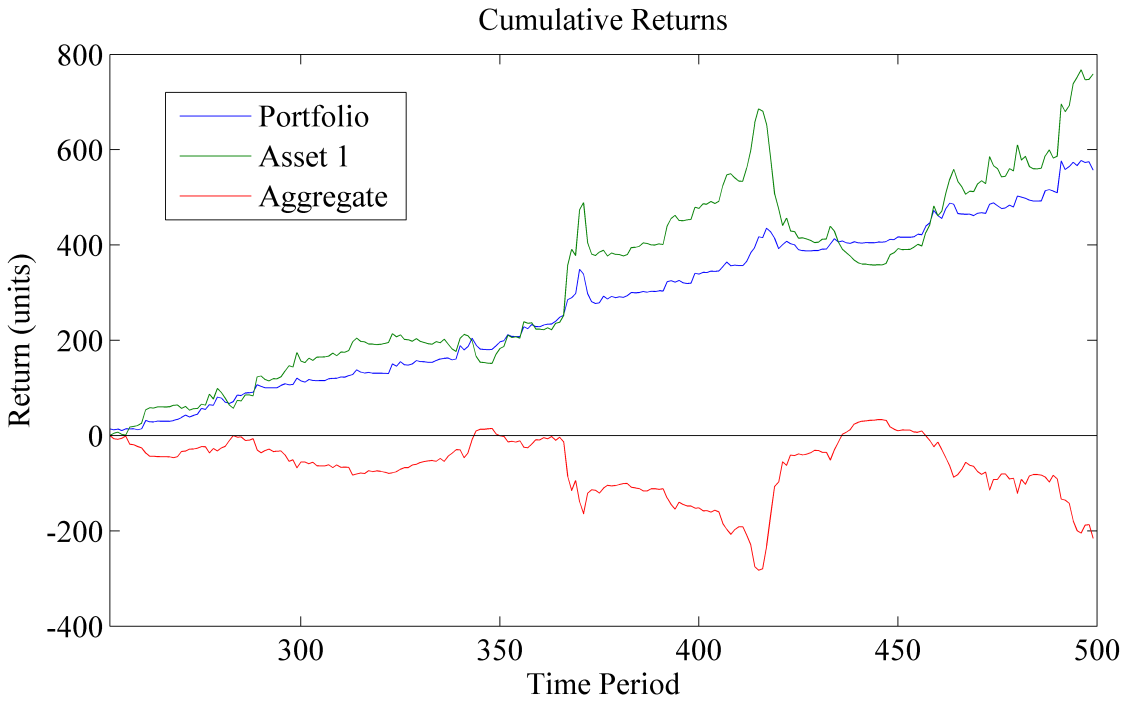


Figure 4.3.6: Artificial Market: Asset and Portfolio Profit Curves

Although we were able to directly observe that the network relationship's estimation of an artificial asset contained significant information about its composition, the pairs trading approach similarly confirmed the model's out-of-sample validity without requiring knowledge of asset's compositions. This makes the algorithmic trading approach a valuable test when applying the network model to real financial markets.

4.4 Discussion

In a transparent market where factors contributing to the construction of assets were linear, constant, and known, the proposed sensor creation layer effectively uncovered information relating to inter-asset associations resulting from shared pricing factors. The process of iteratively sampling randomly selected assets from the market as regressors in linear models was shown to be a sufficient tool to estimate a noisy sensors.

The sensor aggregation technique applied to generate network relationships and

‘aggregate assets’ was bootstrap aggregating. The simplicity of the approach allowed for rapidly implementation to determine if the first layer of the network model had identified structure among assets. One of the shortcomings of the simple aggregation technique was apparent in figure 4.3.3. Note that every factor which received an allocation greater than 10% was consistently underestimated by the network relationship. The underestimation resulted the aggregation methods inability to distinguish among sensor quality and recombine them accordingly to most accurately estimate the target asset.

Addressing this shortcoming is the basis for both the sensor evaluation and sensor processing layers.

People are seduced by signals from the world, but that is manipulation, not reality. Computers have learned more about us than we've learned about them.

- Douglas Rushkoff

5

Sensor Evaluation: Layer 2

5.1 Aims

The sensor creation layer ends with the production of an arbitrarily large number of low quality sensors containing small amounts of information about asset inter-relationships. The sensor evaluation layer aims to evaluate the usefulness, and information overlap, of those sensors.

As discussed previously, there is no convenient method to confirm or refute the validity of the inter-asset relationships implied by sensor weights. Therefore, to determine if the sensors are identifying meaningful structure among assets they must be applied to a problem. In this case the sensors' expectations are collected under varying market conditions and these expectations are compared to actual outcomes. Each sensor is expected to contain a small amount of information relating to inter-asset relationships as a proxy for mutually shared dependence on external market factors. Given the possibility that the average identified structure is not necessarily most representative of the modelled asset, a mechanism for comparing the diversity of information content for sensor was needed.

The only criterion sensors were evaluated on was if identifiable in-sample structure

could be found between their predictions and market realities. This approach is applied to identify information content, not create any expectation as to out-of-sample value.

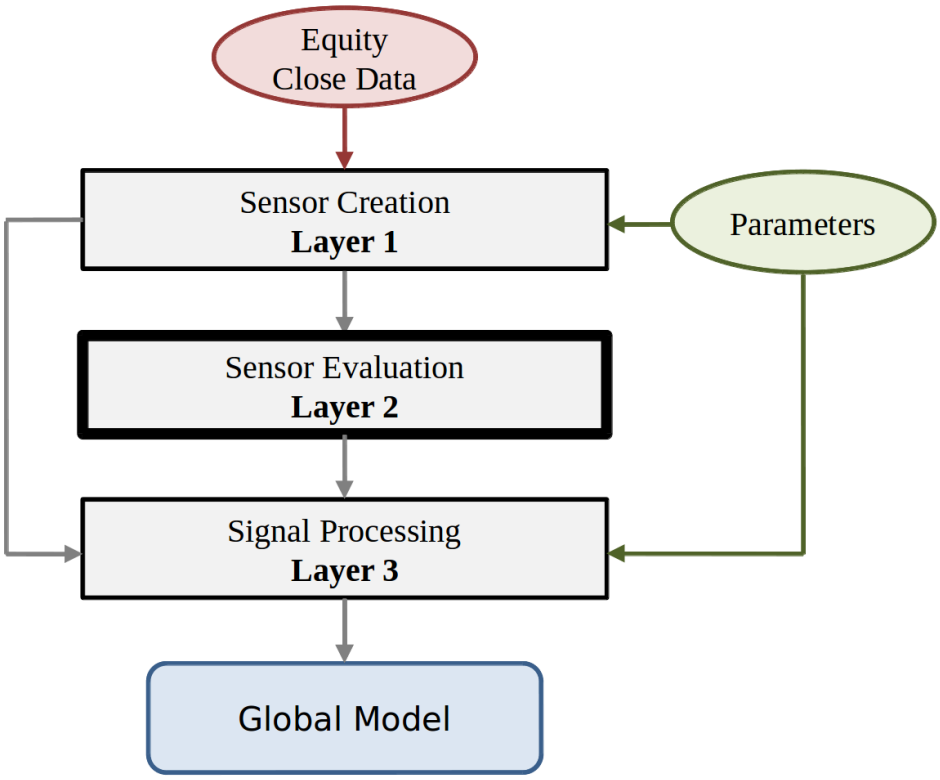


Figure 5.1.1: Sensor Evaluation: Level Flowchart

5.2 Method

A quantitative approach to comparing sensor predictions to market realities was required. The selected method was determining each sensors' profitability when applying a simple in-sample trading strategy on their expectations. This method was selected for 4 reasons:

1. Accessible
2. Minimal Data Requirements
3. Parameterless
4. Continuity

The simulated algorithmic trading approach to determining if sensors are identifying structure is accessible because it was already introduced in the sensor creation chapter. Furthermore, it results in a straightforward profitability curve and associated Sharpe ratio. Several simplifications were made to eliminate the need for external data or additional parameters. This was achieved by ignoring intra-day effects, liquidity, slippage and allowing a decimalized lot size[1] without penalty.

The use of simulated algorithmic trading as a validation of the network model's intermediary steps is a recurring theme in the frameworks creation. This allows for a common language throughout the work and provides a natural progression towards one of the upcoming comprehensive validations.

5.2.1 Notation

Variable	Description	Sample Value
c	a constant term	7.84
ϵ	Error of the model	Time-Series
N	Number of assets in data set	100
L	Number of observations/period per asset	252
w	Number of sensors per target asset	5000
C	Matrix of Daily Close Prices	N by L matrix
T	Index of target asset C	Time-Series
X	Table of sensors	Table [w by N+1]
$\hat{\zeta}$	Vector of normalized sensors signals/residuals	Vector
σ	Covariance matrix of sensor expected information content	Matrix [w by w]
$\bar{z}$	Vector of sensor cumulative returns of length w	Vector [1 by w]

5.2.2 Process

The sensor evaluation layer accepts the sensor table, X, from the sensor creation layer and produces in-sample expected returns and covariances of their information content. These values are then leveraged in the sensor processing layer to aggregate sensors to create out-of-sample predictability in excess of the simple bootstrap aggregating approach previously discussed. Pseudo code identical to the exert in Chapter 3 can be found below.

[1]Portions of shares may be traded

Algorithm 13 Network Model: Sensor Evaluation

procedure EVALUATESENSORS

$w \leftarrow$ Number of sensors per target asset

$N \leftarrow$ Number of assets in data set

$L \leftarrow$ Number of observations/period per asset

$C \leftarrow$ Daily close data matrix [N by L]

$T \leftarrow$ Index of target asset

$X \leftarrow$ Table of sensors

$R \leftarrow$ Matrix of in-sample sensor returns series [w by L-1]

$\hat{\zeta} \leftarrow$ Vector of normalized sensor signals/residuals

$\bar{z} \leftarrow$ Vector of sensor cumulative returns of length w

$\sigma \leftarrow$ Covariance matrix of sensor expected information content [w by w]

top:

$T = 1$

for $k = 1 : w$ **do**

$sensor = X(k, :)$

$targetseries = C(T, :)$

$\zeta = sensor(1 : N) * C + sensor(N + 1)$

$\hat{\zeta} = -\zeta / \mathbf{max}(|\zeta|) \leftarrow$ Residuals normalized to bound -1 to 1

$Tret = 1$ **period difference of** $targetseries$

$r = (1$ **period difference of** $\hat{\zeta}) * Tret \leftarrow$ Daily return from trading

$\bar{z}(k) = \mathbf{sum}(r) \leftarrow$ Cumulative return from frictionless trading

$R(k, :) = r$

for $i = 1 : N$ **do**

$\sigma =$ **Covariances among rows in** $R \leftarrow$ Covariance of sensor returns

The sensor evaluation layer begins by selecting an asset to be modelled, or target asset, T. In the given example the selected asset is Asset 1, i.e. the asset indexed at row 1 in the matrix of close prices, C. The process then multiplies the table of sensors for the target asset with the in-sample daily close matrix of asset prices. The result, ζ, represents the error term of the sensor's cointegration equation.

The error term is the most critical component in the algorithmic trading evaluation because it is the source of future asset price movements expectations. Each sensor's cointegration equation was expected, by definition, to represent a mean-reverting relationship. Therefore, when the error term in the equation is large and positive, the

target asset's price is expected to increase to bring the relationship back into alignment. Conversely, when the error term is large and negative, the target asset's price is expected to increase. This makes the error term a 'signal' for trading decisions.

The ζ term is viewed as a 'raw' signal because it is has no upper or lower bound. Normalizing the values in the time-series results in $\hat{\zeta}$, a signal bound between 1 and -1. This is achieved by dividing the time series by the largest magnitude error term during the in-sample data.

Multiplying the 1 period difference of the signal with the 1 period differences of the target asset, i.e. changes in signal with returns of target asset, creates a daily return from holding the signalled portion of each asset at each time instance. Summing this time series returns $\bar{z}$ or the cumulative sum of returns from frictionless trading. Trading frictions were not included in the sensor evaluation layer because the aim was to identify and discriminate among structure in various sensors. The method's similarity to a trading strategy is not relevant and adding frictions could corrupt the of evaluation metrics.

The 2 outputs passed forward to the sensor processing layer are σ and $\bar{z}$ representing the covariance among sensors information content and expected returns from trading. The model was trained and tested on the same data set so it is known that the information content and expected returns are both overstated. It was expected that the relative levels of covariance among the information content would hold along with the relative levels of expected returns.

5.3 Validation

The sensor evaluation layer does not yield outputs which can be validated. In essence the validation of the sensor evaluation layer occurs during the sensor processing layer. The following validation applies the algorithmic trading component of the sensor evaluation layer to illustrate certain aspects of the outputs. This also provides an opportunity to test the incomplete framework on a real market data to determine if the intermediate stage detects structure, and out-of-sample predictability in the same way as the artificial market. As outlined in algorithm 13, the validation compares predictions made possible by the sensors to market realities and reviewing discrepancies. The comparison is straightforward as the predictions indicated by sensors lead directly to trading decisions. The sensors' ability to discover meaningful long-run trends in inter-asset dynamics is evaluated by observing the out-of-sample

profitability of their implied trading decisions.

Data

The data utilized in the unhedged market intermediary validation were the 100 common stocks highlighted in figure 5.3.1. The remaining 313 drawn from the S&P 500 index meeting the criteria outlined in chapter 3 are also shown and were used in later chapters. A shortened backtesting period was selected for the validation encompassing January 2005 to March of 2008. The data set was limited to this range to avoid potentially compounding a data snooping effect which could bias later tests. The data frequencies were daily with prices captured at market close.

Simulated Algorithmic Trading

While all 413 assets were allowed as potential regressors in the sensors, only the highlighted 100 assets were accepted as target assets. As in the artificial market, random selections of the global 413 assets were regressed onto each of the 100 target assets resulting in a table of sensors. As described in algorithm 13, the table of sensors enabled computation of an in-sample trading signals. The final trading signal for each sensor contained no future data and was expected to have value out-of-sample. Averaging the signals, i.e. bootstrap aggregating, led to the out-of-sample traded signal for each target asset.

In the artificial market, 1 target asset was observed and extensively analysed. In the coming example, a more global approach was demonstrated. The described process was performed on each of the 100 listed target assets. Outcomes for all assets are outlined aiming to identify any recurring trends and eliminate a possible source of selection bias e.g. selecting an asset which performs as expected.

Predictability, i.e. simulated profitability, was measured in dollar units and Sharpe ratios where a normalized signal of 1 creates a $1 investment in the relevant target asset. Simulated asset holdings and profits/losses were realigned daily according to the sensors' signals.

This scheme measures returns by changes in the portfolio's value as opposed to realized cash gains i.e. mark-to-market. Using the portfolio value as returns grants clarity to the overall performance at each time instance. Plotting the cumulative sum of each asset's returns created the profit curves found in figure 5.3.2.

Equity Ticker Symbols									
A	AA	AAPL	ABC	ABT	ACAS	ACE	ADBE	ADI	ADM
ADP	ADSK	AEE	AEP	AES	AET	AFL	AGN	AIG	AIV
AIZ	AKAM	ALL	ALTR	AMAT	AMD	AMGN	AMT	AMZN	AN
ANF	APA	APC	APD	APOL	ASH	ATI	AVB	AVP	AVY
AXP	AZO	BA	BAC	BAX	BBBY	BBT	BBY	BC	BCR
BDX	BEN	BF.B	BHI	BIG	BIIB	BK	BLL	BMC	BMS
BMY	BRCM	BSX	BTU	BXP	C	CA	CAH	CAT	CB
CBE	CBG	CCE	CCL	CCU	CELG	CHK	CHRW	CI	CIEN
CINF	CL	CLX	CMA	CMCSA	CME	CMI	CMS	CNP	CNX
COF	COH	COL	COP	COST	CPB	CPWR	CSC	CSCO	CSX
CTAS	CTL	CTSH	CTXS	CVG	CVH	CVS	CVX	D	DD
DDS	DE	DELL	DF	DGX	DHI	DHR	DIS	DOV	DOW
DRI	DTE	DTV	DUK	DVN	EBAY	ECL	ED	EFX	EIX
EL	EMC	EMN	EMR	EOG	EQR	ESRX	ESV	ETFC	ETR
EXC	EXPD	F	FCX	FDO	FDX	FE	FHN	FII	FIS
FISV	FITB	FLR	FRX	GAS	GCI	GD	GE	GILD	GIS
GLW	GME	GNW	GOOG	GPC	GPS	GS	GT	GWW	HAL
HAR	HAS	HBAN	HCBK	HD	HES	HIG	HNZ	HOG	HON
HOT	HPQ	HRB	HSP	HST	HSY	HUM	IACI	IBM	IFF
IGT	INTC	INTU	IP	IPG	IR	ITT	ITW	JBL	JCI
JCP	JDSU	JEC	JNJ	JNPR	JNS	JNY	JPM	JWN	K
KBH	KEY	KFT	KIM	KLAC	KMB	KO	KR	KSS	LEG
LEN	LH	LLL	LLTC	LLY	LM	LMT	LNC	LOW	LSI
LTD	LUK	LUV	LXK	M	MAR	MAS	MAT	MBI	MCD
MCHP	MCK	MCO	MDP	MDT	MET	MHP	MIL	MKC	MMC
MMM	MNST	MO	MOLX	MON	MRK	MRO	MS	MSFT	MTB
MTG	MTW	MU	MUR	MWV	MYL	NBL	NBR	NE	NEM
NI	NKE	NOC	NOV	NSC	NTAP	NTRS	NUE	NVDA	NWL
NYT	NYX	ODP	OMC	OMX	ORCL	OXY	PAYX	PBI	PCAR
PCG	PCL	PCP	PDCO	PEP	PFE	PFG	PG	PGR	PH
PHM	PKI	PLD	PLL	PNC	PNW	POM	PPG	PPL	PRU
PSA	PX	QCOM	QLGC	R	RAI	RDC	RF	RHI	RIG
RL	ROK	RRC	RRD	RSH	RTN	S	SBUX	SCHW	SEE
SHLD	SHW	SIAL	SLB	SLM	SNA	SNDK	SO	SPG	SPLS
SRE	SSP	STI	STJ	STR	STT	STZ	SUN	SVU	SWK
SWY	SYK	SYMC	SYY	T	TAP	TE	TEG	TER	TEX
TGT	THC	TIE	TIF	TJX	TLAB	TMK	TMO	TROW	TRV
TSN	TSO	TSS	TWX	TXN	TXT	TYC	UIS	UNH	UNM
UNP	UPS	USB	UTX	VAR	VFC	VLO	VMC	VRSN	VZ
WAG	WAT	WEN	WFC	WFR	WFT	WHR	WLP	WM	WMB
WMT	WPI	WPO	WY	X	XEL	XL	XLNX	XOM	XRX
YHOO	YUM	ZION							

Figure 5.3.1: Single Asset: Ticker Symbols

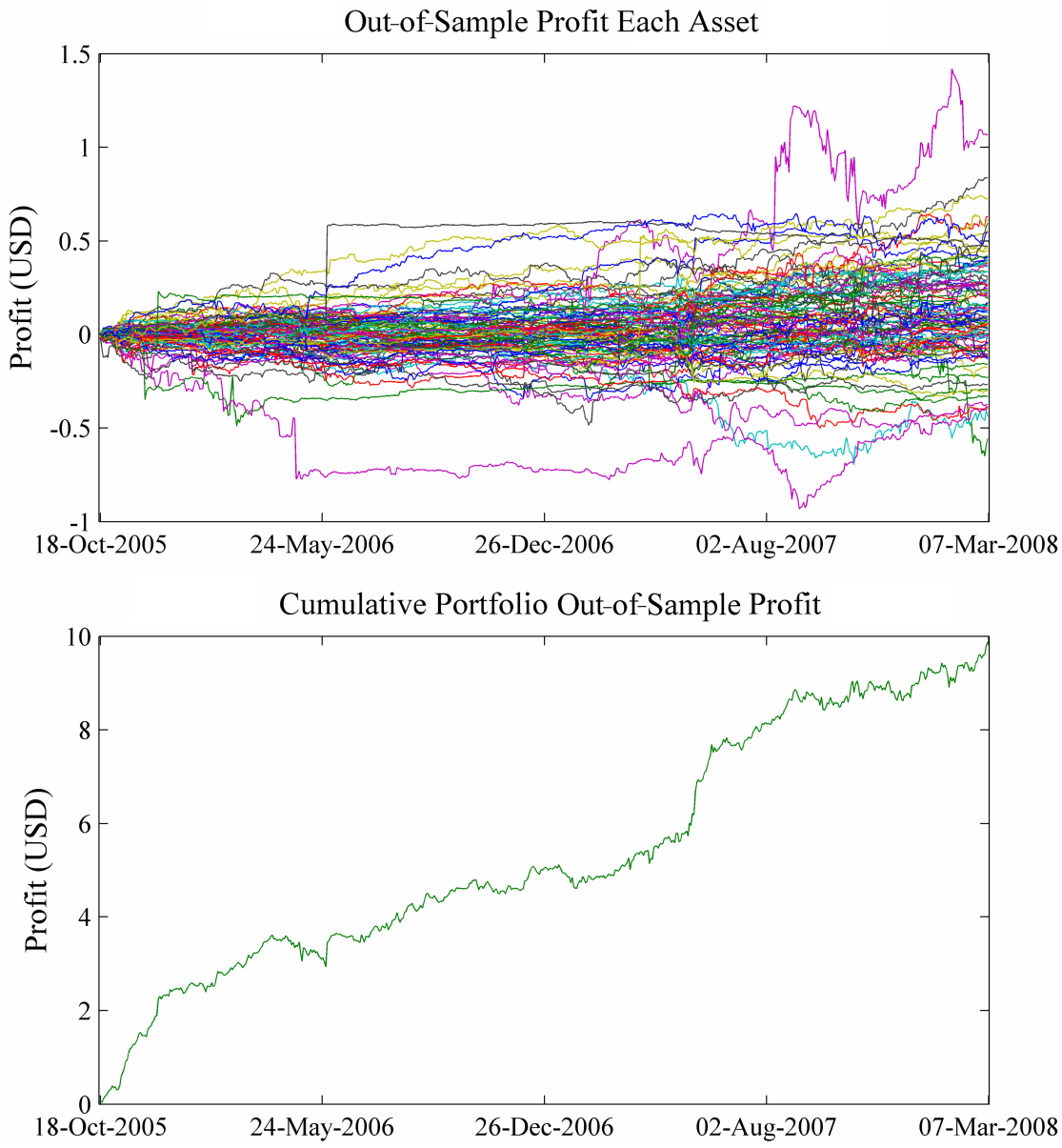


Figure 5.3.2: Single Asset: Normalized Price Profit Curves

The individual simulated profit curves for selected assets do not directly show a clear tendency for sensor predictability/profit. There are instances of immediate and consistent profits and losses in various assets. Review of the cumulative portfolio profit curve was a more decisive. The simulated strategy generated $9.78 of profitability over the evaluation period. The exact value of performance is not relevant beyond that it is consistent and upward trending. This consistent positive trend was reaffirmed by the simulated strategies Sharpe ratio of 2.37.

Using the least sophisticated of signal processing technique, i.e. averaging, 100 assets, and no consideration for market neutrality, the results indicate that sensors identified structure among assets.

5.4 Discussion

Results from simulating out-of-sample trading decisions based on a simple average of sensors' signals showed that sensors' expectations contained some degree of information relating to future market movements. This was demonstrated by the positive and consistent simulated profitability when combining returns from all assets.

The positive simulated profitability does not imply that the trading strategy is valid for real world use. The predictability only determines that sensor creation layer generated sensors identifying some degree of long-term (out-of-sample) structure among assets. This validation supports the hypothesis and several assumptions necessary to the frameworks structure. Specifically:

1. Market externalities impact price
2. Assets share dependencies of market externalities
3. Asset external dependencies change slowly

These assumptions can not be proven outright. Notwithstanding, the results from sensor evaluation align with expectations in the paradigm where external factors may be shared and have a meaningful impact on asset pricing.

The approach to constructing an overall model, a trading signal in this case, of inter-asset relationships by averaging sensor models is flawed. Thinking in terms of the shared market factor pricing framework this method introduces a bias. There is no way of determining the overall distribution of factor reliance among assets in the

market. For this reason, it is likely that a simple average would overstate the relative importance of a shared factors that are relevant to many assets. In practical terms this could mean an asset heavily reliant an obscure factor, e.g. availability of ceramic, would be understated relative to a more globally relevant factor, e.g. the LIBOR rate.

The final outputs of the sensor evaluation layer are sensor information content, estimated by in-sample predictability, and overlap in information content, estimated by a covariance matrix of in-sample predictability. These metrics are passed forward to the sensor processing layer to improve the issue of common factors being overstated in each network relationship.

It is expected that trading the simulated portfolio based purely on average sensor signal appeared so predictable was partially because signals were somewhat naturally self-hedging. Assuming the number of buy and sell signals in the portfolio tended to be equal the strategy would remain roughly cash neutral. While cash neutrality was in no way relevant to the algorithms predictability in the market, it may aid in insulating the portfolio from overall market movements. In other words, if the sub-portfolios of long and short positions both tended to mirror global market movements, the portfolio as a whole would have naturally emergent considerations for market neutrality.

The sensor evaluation layer has shown that sensors identified meaningful structure pertaining to inter-asset relationships. The simplistic approach to creating a single model/signal for each target asset was functional. It was believed that post-processing the signals could allow identification of over/under represented market factors and enable increasing/decreasing their impact respectively. One approach to improving the overall model of assets is developed in the sensor processing layer of the market modelling framework.

5.5 Implementation

Implementing the network model with a large numbers of assets and/or sensors rapidly becomes computationally infeasible. For best results, each asset in the data set is modelled by thousands of sensors, every 1 of which is computationally demanding. The computational requirements are similarly impacted by parametrization. For example, increasing the number of regressors per sensor, sensors per asset, or the training data period all magnify resource requirements.

The nature of the framework does allow for a notable amount of parallelism. The

sensor creation layer has a duality of parallelism in that for a single asset, all sensors may be computed independently and all code for each asset may be run asynchronously. Both constructs fall into the classification of single instruction multiple data or SIMD.

Data

The discussed framework was most tractable to equities markets. Knowing this, the first decision was from which exchange to draw target common stocks. The New York Stock Exchange (NYSE) was the clear front-runner with the largest market capitalization compared to its counterparts. The S&P 500 index was selected as the benchmark for comparison.

Drawing from the S&P 500, all available equity close prices were selected over the range from January 3^{rd}, 2005 to January 8^{th}, 2014. These dates also represent the first trading day in 2005 and the date of download in 2014. This period encompassed the growth period from $2005 - 2007$, the financial recession in 2008, and partial recovery through 2014. The aim of using such a diverse data set was to determine if the modelling framework performed differently under varying market states. All data was drawn from Google Finance[2] for historic common stock prices. Following the selection, data were filtered to include only assets which contained no sparsity over the selected date range. All data were sampled at daily frequency.

Upon filtering, there were 413 remaining assets. Filtering data on sufficient history eliminates companies which fail over the evaluation period. This form of snooping on future data is known as a survivorship bias[15] and can make algorithms appear more predictable than they are. While it would be possible to leave failing instruments in the pool, doing so would further increase the computational requirements. For this reason, they were omitted. The ticker symbols of the remaining 413 common stocks can be found in figure 5.3.1.

Computations

The proposed network model of markets is computationally expensive. In the artificial market validation from the previous chapter solving the global model for a single target asset involved $5,000$ sensors each with 8 regressors and a constant. Each sensor required simultaneously solving a system of equations which, on its own, is a

[2] A Google Inc web service providing business/financial news and financial instrument data

CPU cycle intensive task. For this reason, the usual statistical packages and programming languages associated with statistical tasks such as EViews[3], Microsoft Excel[4], R[5], and MATLAB[6] were all infeasible.

The goal of the selected approach was to maximize execution speed. The highly parallel nature of the process suggested that the implementation would take advantage of the single instruction multiple data paradigm present at each stage of analysis. General purpose graphics processing unit (GPGPU) computing is intended precisely for this purpose.

Graphics Processing Units (GPUs), initially designed for graphics rasterisation, are increasingly receiving attention in the scientific computing community. This interest spurs from the massive raw compute capabilities associated with GPUs' highly parallel architecture. The subsequent advent of general purpose GPU computing languages such as Compute Unified Device Architecture (CUDA) and Open Compute Language (OpenCL) have made GPU acceleration of computationally expensive parallel algorithms an option. General purpose GPU programming remains in its infancy. Given this is the target hardware for the proposed algorithms we first discuss the internal architecture and limitations of the GPU.

Our discussion of GPUs is limited to the nVidia Kepler architecture present on the nVidia GeForce 660 Ti[7] utilized for processing. The maximum programming capability of the 660 Ti is outlined by CUDA 5[8]. The 660 Ti contains 7 streaming multiprocessors (SMXs), each of which houses 192 CUDA cores leading to a total number of $1,344$ CUDA cores. Each of these computational units is clocked to 980 MHz. The host system for this device contained an quad-core Intel i7 processor clocked at 3.07 GHz. Although the comparison is meaningless for all but perfectly parallel and data-less tasks, the GPU's combined computational power of $1,317.1$ GHz versus the CPU's 12.3 GHz hints at the potential for speed-ups in scientific computing.

Each of the 7 SMXs is physically isolated on chip which prevents their cooperation. Within each SMX however, the user may assign individual cores to programmatic

[3]Statistical modelling and simulation application

[4]An interactive virtual spreadsheet application with built in Visual Basic interpreter interoperability

[5]An open source programming language designed for statistical computing

[6]A closed source programming language and environment specializing in numerical computation and visualization

[7]Upper-mid consumer grade graphics card

[8]2012 release of nVidia's GPU programming model

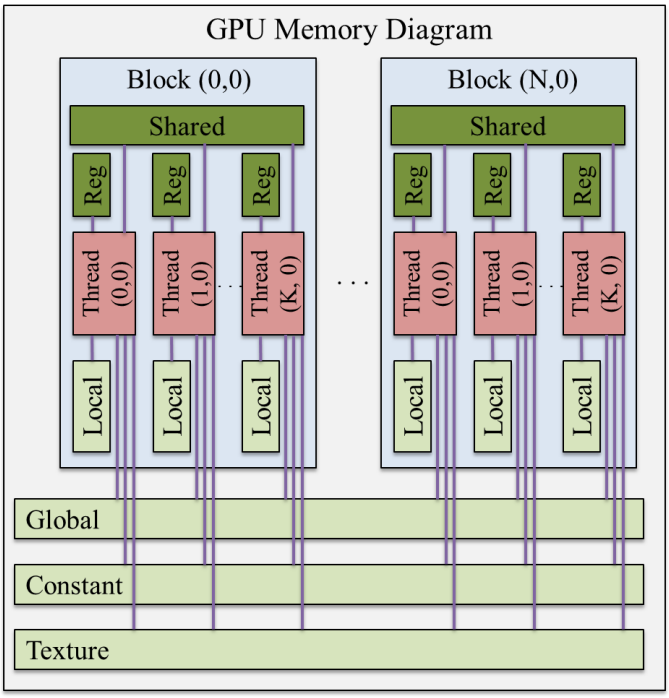


Figure 5.5.1: Single Asset: GPU Memory Diagram

constructs known as blocks. Within a block all continuous execution units are known as threads. Every thread within a block has the potential for group cooperation through high speed shared memory and thread synchronization with slight overhead. Synchronizing among blocks both on and off the same SMX incurs significant overhead and is not directly supported. There are several notable limitations when allocating blocks of threads to perform work. As of CUDA 5, no block may contain more than 2048 threads. Also, all tasks are issued in 32 thread increments known as Warps.

Best practices for GPU computing do deviate from those of CPU computing. Most significant of these is the need to avoid logical statements in favor of launching additional threads. Launching threads requires little overhead in comparison to the equivalent CPU task. It is common practice to launch a series of threads for the sole purpose of performing one time tasks as trivial as summing a vector. This approach nearly always preferable to logical operations or loops which are relatively slow. The memory layout present on GPUs also leads to some considerations which violate CPU standard practices.

From CPU memory, data can be pushed to global, constant and texture memory all of which are located on the GPU. The speed of this transaction can be limited by

either the card's memory bus or the host system motherboard PCI-E port. The maximum speeds of PCI-E versions 1, 2, and 3, are 4, 8, and 16 GB/s respectively for 16-lane slots.

Threads within blocks have access to 6 memory types all with unique properties. Global memory or DRAM is the largest bank with 2 GB on the 600 Ti. The current maximum amount of global memory on any consumer grade nVidia GPU is 6 GB. Global memory is located off chip and resultantly has the highest access latency of any memory type. Registers, in contrast, have thread scope and are limited to 63 32-bit registers per thread. A further limitation on registers is that access locations must be available at compile time. It is not possible to variably access register addressable memory. For this reason, and to provide spillover for over addressed registers, local memory exists as a programmatic construct. Local memory is simply global memory with thread scope. A more interesting option for many of the proposed framework's purposes is shared memory. Shared memory has block scope but is limited to 48 KB. Block scope enables all allocated threads in the block to interact and synchronize while performing tasks located in this high speed section. This opens the possibility for utilizing all $1,344$ cores to perform fewer than $1,344$ independently parellelizable tasks without incurring significant reductions in realized performance. The access latency difference between shared and register memory vs global and local memory is on the order of 100 times. Finally, constant and texture memory are unused in implementation and are omitted from the section.

The GPU architecture was extremely well suited to accelerating the sensor creation layer. To exploit the GPU a block parallel structure was adopted. Given the knowledge that 252 trading days was the training data length and the number of regressors was not to exceed 20 in any tests, 48 KB available in block shared memory was sufficient to house the necessary data for any single sensor. Once data was effectively pushed to the GPU an estimation of the population parameters minimizing square errors with Gaussian residuals was computed with the aid of the cuBLAS[9] library. The sensor data was then routed back to the local system's hard drive (HDD) for further analysis.

[9]NVIDIA supported CUDA accelerated linear algebra library.

One of the funny things about the stock market is that every time one person buys, another sells, and both think they are astute.

- William Feather

6

Sensor Processing: Layer 3

6.1 Aims

The sensor processing chapter introduces a technique aiming to alleviate several outlined shortcomings in the existing approach to modelling financial markets. The previously demonstrated approach to modelling an asset consists of creating thousands of sensors and then averaging these sensors together to create a aggregate model. Given the linear model approach to generating the noisy sensor observations there were several known concerning features.

The sensor evaluation layer showed that sensors are capable of identifying meaningful structure pertaining to inter-asset relationships. The simple approach to creating a single model for each target asset was able to generate measurable out-of-sample predictability. It was believed that post-processing the sensors could allow identification of over/under represented market factors and enable decreasing/increasing their impact respectively. One option for improving the overall model of assets is developed in the sensor processing layer of the modelling framework.

The most notable issue associated with averaging sensor models we attempt to overcome is the known misrepresentation of each asset's factor composition. Since regressors in sensors are randomly selected from the pool of all assets, if certain factors are over-represented in data set, they are likely to be over-represented in sensors. The signal processing layer aims to analyse sensors and combine them in a way that improves accuracy of assets' factor compositions.

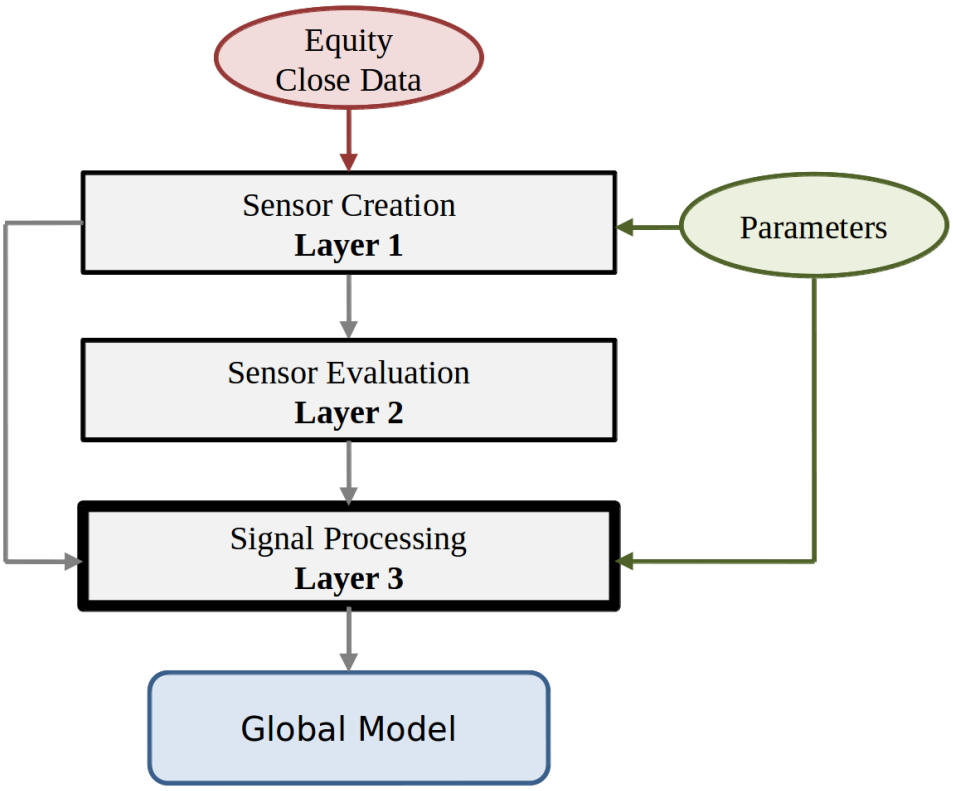


Figure 6.1.1: Sensor Processing: Level Flowchart

6.2 Method

In this chapter we identify an alternate method for recombining sensors to produce more accurate models of asset inter-relationships and factor compositions. In short, mean-variance analysis (MVA) is the technique used to distinguish if structure identified by sensors is shared or unique. From this information it is trivial to construct a 'portfolio' of sensors which are expected to be a superior representation of

assets' inter-relationships and factor compositions when compared to a simple average.

While MVA is typically associated with portfolio optimisation, it is also a useful signal processing tool. MVA provides a mechanism to diversify the origin of identified structure among sensors. Applying this technique to sensors identifies a weighted combination of sensors which more accurately represents an asset's interrelationships and factor composition. The specific aspects of MVA required for the signal processing technique are presented in the remainder of the section. More general information is available in chapter 2.

6.2.1 Notation

Variable	Description	Sample Value
N	Number of assets in data set	100
σ	Covariance matrix of sensor expected information content	Matrix [w by w]
$\bar{z}$	Vector of sensor cumulative returns of length w	Vector [1 by w]
δ	Number of sensors per sensor portfolio	20
w	Number of sensors per target asset	5000

6.2.2 Process

Mean-variance analysis is discussed at length in the background and literature review chapter. The two properties necessary for MVA are, a series of historic returns which takes the place of expected returns, and the covariances among those returns. The sensor evaluation layer ended with the creation of a σ matrix and $\bar{z}$ vector. These variables contain the matrix of all sensor's return series covariances and their expected returns respectively.

The goal of the process is to recombine sets of sensors into sensor portfolios which diversify the source of information content while maintaining an efficiently predictive model. Diversifying the source of information content allows frequently identified structure to be reduced in weight and information which is infrequently identified, but equally predictive, to be overweighted. The process aimed to more accurately represent the underlying factor compositions of assets thereby improving the model.

Algorithm 14 first splits sensors into equally sized groups defined by δ. It then computes the mean-variance optimal recombination weights for the subset of sensors given the relevant excerpts of variables from the sensor evaluation layer. The selected

point of recombination was the weights of the point of tangency of a line drawn from the expected risk-free rate, 0.00% in this case, and its single point of contact with the efficient frontier. This point is also the portfolio with the highest expected Sharpe ratio.

Algorithm 14 Network Model: Sensor Processing

procedure SensorProcessing
  $N \leftarrow$ number of assets in data set
  $\delta \leftarrow$ number of sensors per sensor portfolio
  $\sigma \leftarrow$ covariance of returns for sensors $[w$ by $w]$
  $\bar{z} \leftarrow$ returns for sensors $[1$ by $w]$
  $w \leftarrow$ Number of sensors per target asset
  $psensors \leftarrow$ Table containing sensor portfolios $[w/\delta$ by $N+1]$
top:
  $\delta = 20$
  for $i = 1 : w/\delta$ **do**
    $idx = ((i-1)\delta) + 1 : i * \delta$
    $SubSigma = \sigma(idx, idx) \leftarrow$ Excerpt of σ for sensors in current portfolio
    $SubZbar = \bar{z}(idx) \leftarrow$ Excerpt of $\bar{z}$ for sensors in the current portfolio
    $uvec =$ **vector of** 1 **of size** δ
    $OptWts = (SubSigma^{-1} * SubZbar)/(uvec' * SubSigma^{-1} * SubZbar)$
    $psensors(i,:) = OptWts' * X(idx,:)$
  $NetworkRelationship = ($**sum columns of** $X)/(w/\delta)$

After the number of sensors per portfolio, δ, is defined, the main loop divides the total number of sensors for a single target asset, w, into even groups. For convenience, w should be set such that it is evenly divisible by the desired δ. The relevant excerpts of the covariance matrix, σ, and expected returns, $\bar{z}$, are then extracted. The mean-variance optimal weights can then be computed from the extracts and applied to the sensor's weights in the table of sensors, X. This results in an intermediary variable table of portfolio sensors.

The aggregation of portfolio sensors utilizes bootstrap aggregating as previously defined. The values in each sensor portfolio created for a single target asset are averaged resulting in 1 network relationship. Every network relationship represents the final output from the entire process. The network model is a collection of all network relationships, 1 per asset, each of which describes a long run relationship among assets.

6.3 Validation

The mean-variance analysis approach to post-processing sensors into a network relationship superior to a simple average is validated with two experiments. The first validation returns to the artificial market utilized in the sensor creation layer. The experiment compares the aggregate model for an asset created by averaging, and from mean-variance analysis sensor processing. Since the true composition of assets is known in the artificial market, the accuracy of each can be determined. The latter validation involves clustering assets according to their inter-asset dependencies and comparing the identified nearest neighbours to intuitive expectations.

6.3.1 Artificial Market Validation

The artificial market validation returns to an example problem from the sensor creation layer and attempts to improve upon the original solution using sensor processing to create aggregate models for assets. As with the initial approach, added value of mean-variance analysis can be observed in the artificial market.

The experiment from the artificial market validation of the sensor creation layer was the estimation of an artificial asset's known factor composition. The final output comparing the assets true factor composition compared with a simple average of sensors' models can be seen in figure 4.3.3. This output had an cosine angle similarity score of 0.863. We aim to improve this metric using the outlined sensor processing method to construct the factor allocations for artificial asset 1.

During the sensor processing layer, the number of sensors per portfolio, or δ, was set to 20. Sensors were drawn from the table of sensors, X, for asset 1. Each simulated cumulative profit curve from the sensor evaluation layer is seen in figure 6.3.1. Due to the in-sample nature of the training data, highlighted in yellow, all but one sensor produced a positive simulated return series over the first 250 periods. The model's predictability was degraded in the out-of-sample region where 5 of the sensor failed to produce positive returns.

The covariance matrix of simulated returns among sensors in figure 6.3.2 hinted of significant diversity in the factors contributing to each sensor's output. This was not a surprising result as we had prior knowledge that each sensor contained only a small subset of factors from a wide selection of assets. Computing the sensors' returns,

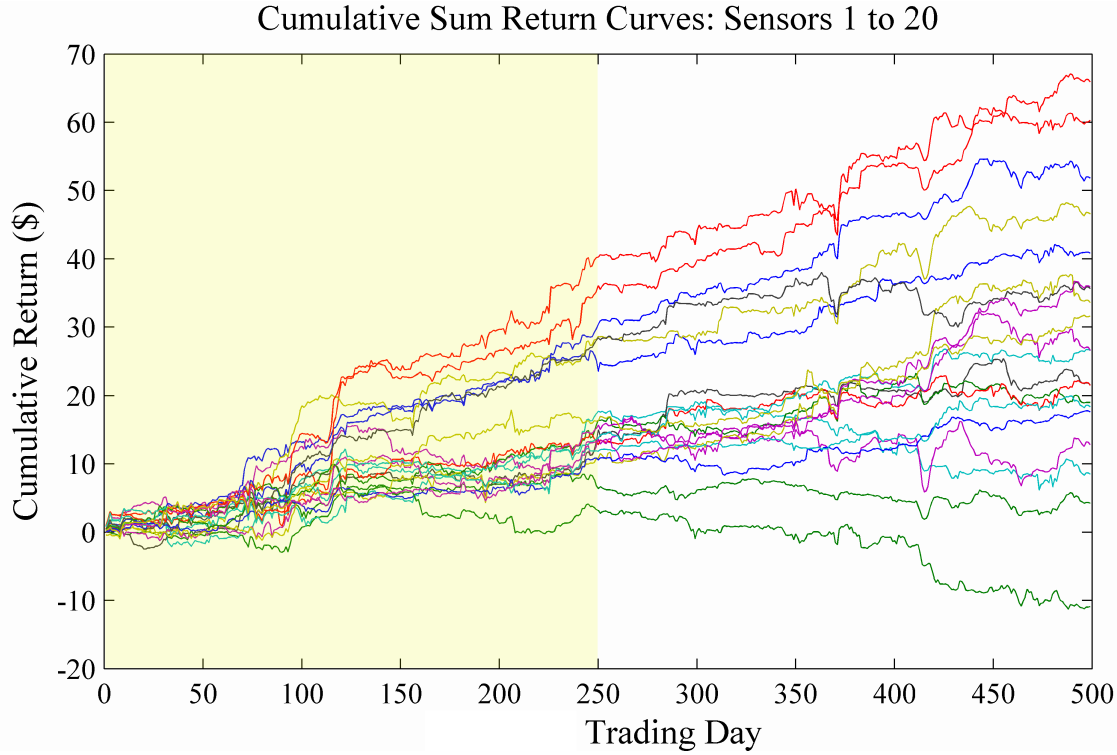


Figure 6.3.1: Sensor Processing: Sensors 1 to 20 Returns

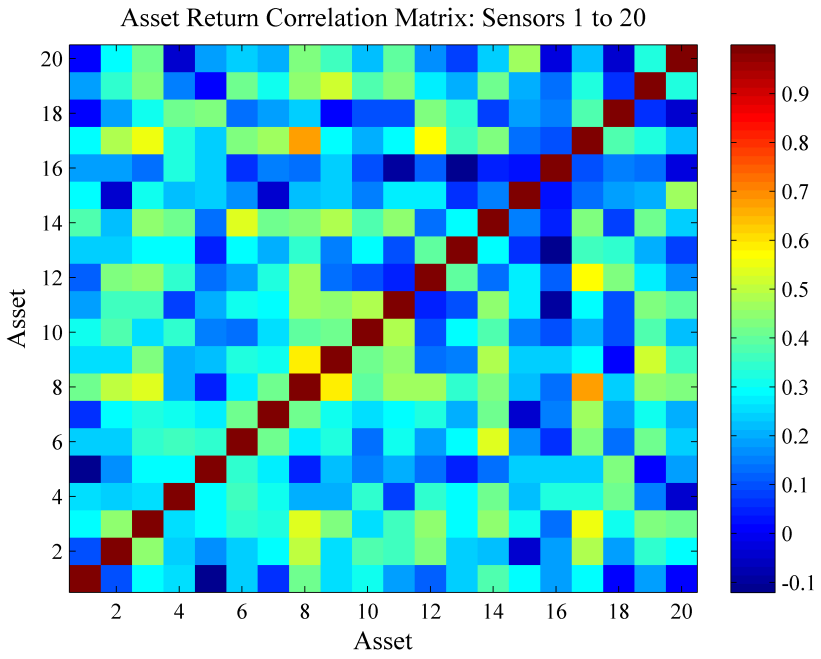


Figure 6.3.2: Sensor Processing: Sensors 1 to 20 Return Correlation

variances, and covariance matrix lead to the efficient frontier of sensor recombination weight point located in figure 6.3.3.

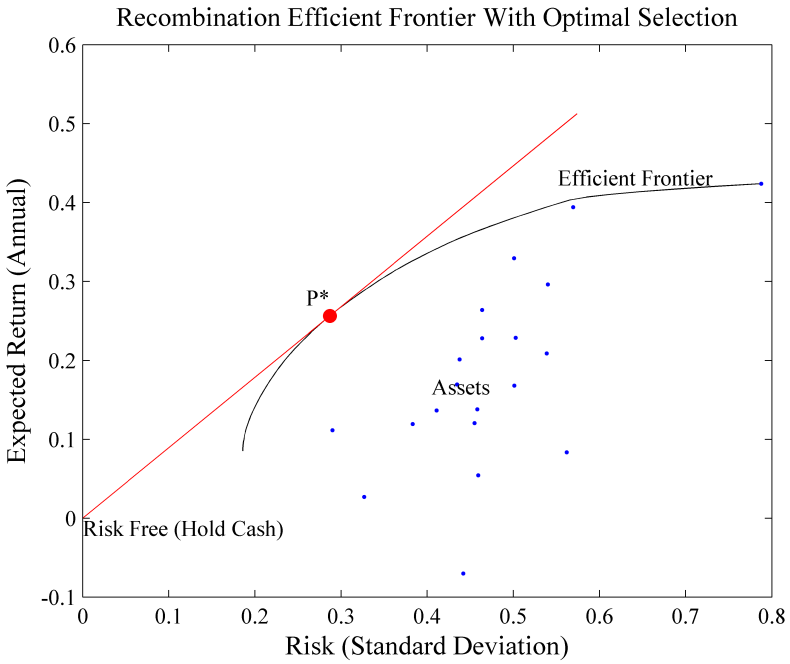


Figure 6.3.3: Sensor Processing: Sensors 1 to 20 Optimal Recombination

The weights for the optimal point on the efficient frontier was found as described in algorithm 14. The selection was made from the perspective of a risk adverse individual with an optional risk-free investment returning 0.00%. The selected recombination point was the point of tangency of a line spanning from the risk-free asset to the efficient frontier[60] and is denoted P^* in figure 6.3.3.

With a point on the efficient frontier, the 20 sensors were then be multiplied by their appropriate weights to produce a new, higher quality sensor portfolio. Repeating the process for the remaining 4980 available sensor models from the artificial market experiment created a total of 250 sensor portfolios. The performance of the portfolio sensors in the same evaluation as figure 6.3.1 presented in figure 6.3.4.

The simulated return series associated for the sensor portfolios were notably more consistent and positive when compared to the individual sensors in figure 6.3.1. The increased simulated profitability was found both in and out-of-sample implying that diversification of information sources was possible, and the analysis improved underlying factor allocations.

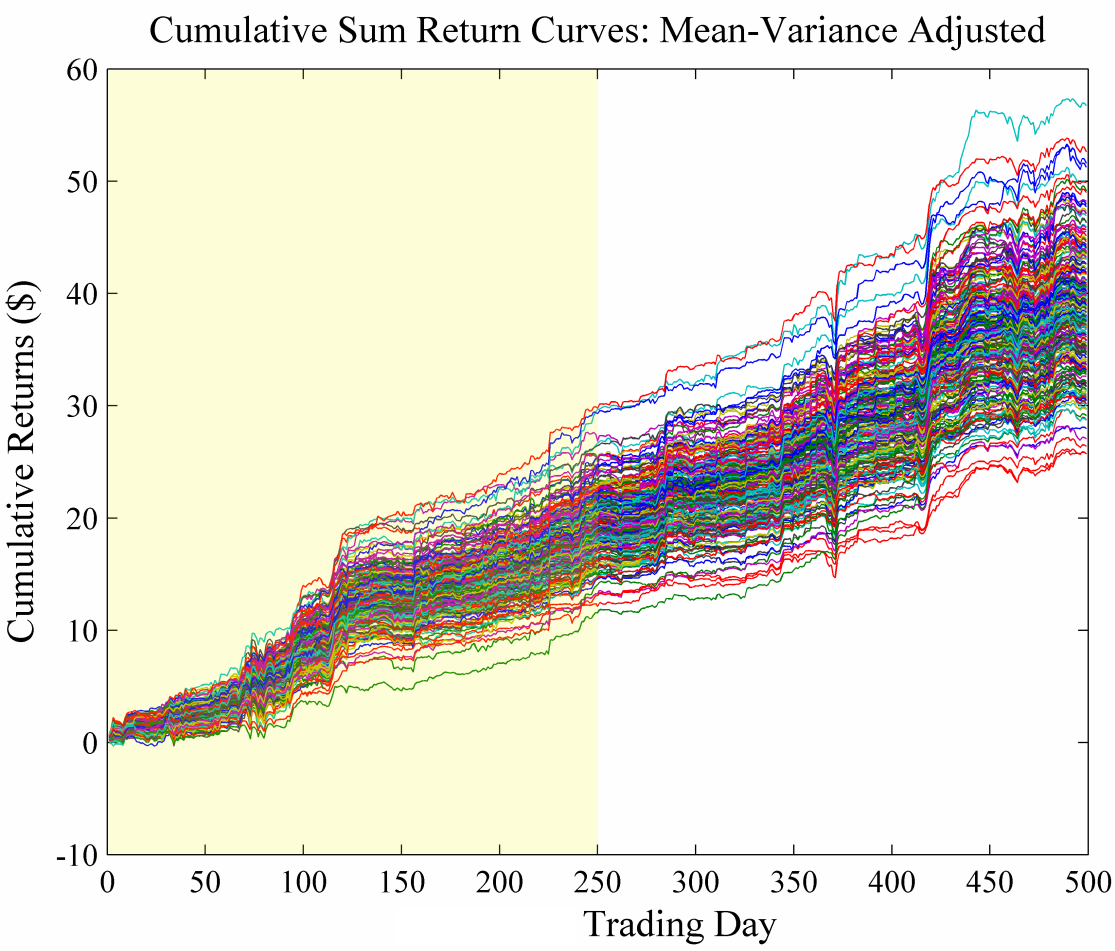


Figure 6.3.4: Sensor Processing: MVA Optimally Recombined Sensor Returns

The hypothesis that sensor processing could improve the estimation of factor composition for each target asset was supported by the results. This hypothesis was further supported by the apparent correlation among processed sensors relative to the single observation series in figure 6.3.3. Correlation of outputs is a beneficial result as it implies that a significant degree of identifiable sources of diversification have been effectively leveraged into improved estimates of the asset's model. Moreover, it suggests that the sensor portfolios are 'converging' to similar models of the target asset.

Despite the positive indications that mean-variance analysis had improved the quality of output sensors, the procedure was only deemed valuable if the factors composing each signal observation led to a more accurate representation of the target asset when adjusted by the portfolio weights. As before, the backtracked factor estimations were rendered in a bar chart. See figure 6.3.5.

In comparison to the equivalent bar chart in figure 4.3.3, every factor containing more than 10% allocation was estimated more accurately after sensor processing when compared to the simple average. This demonstrates that the processing approach is able to adjust sensor weightings to better reflect underlying factor allocations beyond a simple average. Improvements in performance were mirrored by estimates of unused factors which tracked more closely to 0.00 after processing. The cosine angle score increased from 0.863 to 0.961. These results strongly suggests that the sensor processing layer improved the modelled relationships.

6.3.2 Nearest Neighbours Validation

The nearest neighbours validation compares aggregate models created by simple averaging and by mean-variance analysis sensor processing. The starting point for both begins after the sensor creation layer has produced a table of sensors, X. An example excerpt the from table of sensors is visible in table 6.3.1

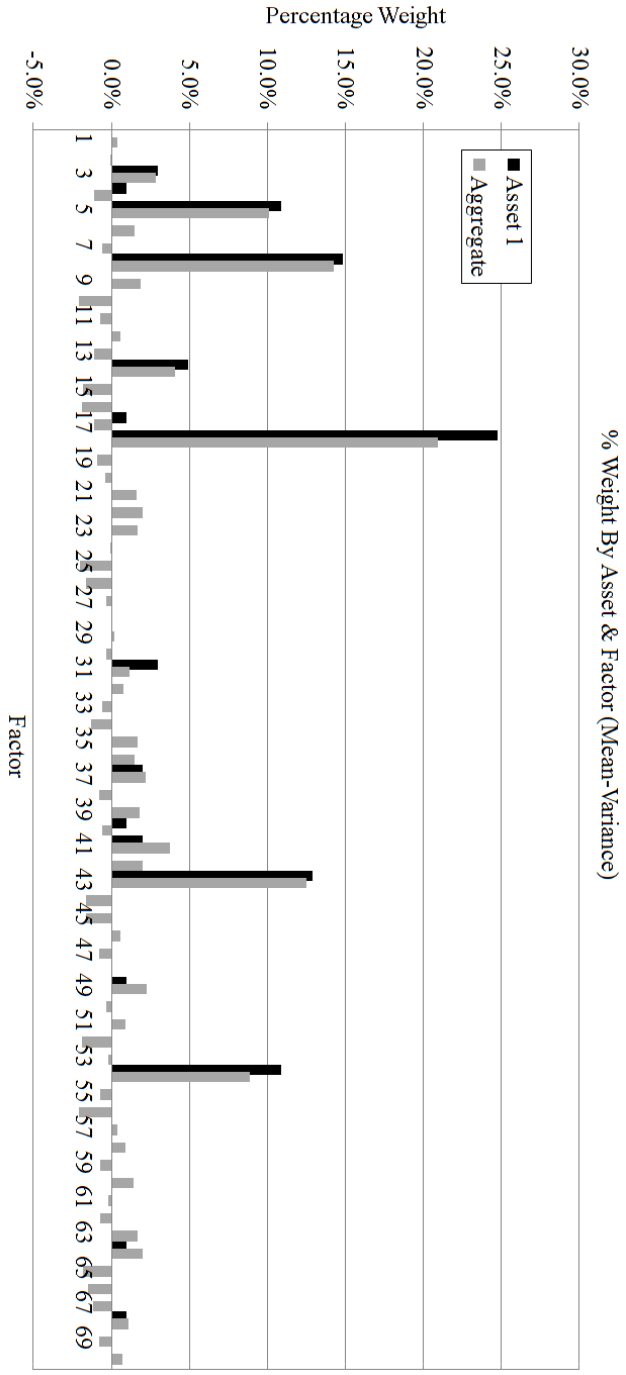


Figure 6.3.5: Sensor Processing: Estimated Factor Weight Allocations

Table 6.3.1 Table of Sensors: Single Target Asset

ϕ_1	ϕ_2	ϕ_3	ϕ_4	ϕ_5	ϕ_6	ϕ_7	ϕ_8	ϕ_9	ϕ_{10}	c
0.09	-1.00	0.00	0.30	-0.36	0.00	0.00	0.00	0.00	0.11	41.38
0.00	-1.00	0.00	0.35	0.00	0.00	0.46	-0.18	0.00	0.11	11.77
0.00	-1.00	0.00	0.28	-0.23	-0.31	0.00	0.00	0.00	0.00	61.97
0.00	-1.00	0.00	0.00	0.00	-0.31	0.25	0.00	0.29	-0.24	50.30
0.05	-1.00	0.00	0.00	0.00	-0.18	0.00	-0.19	0.13	0.00	58.78

From the table of sensors, X, both described processes to define network relationships were applied. First, network relationships for assets were found by averaging the weights of each column, i.e. bootstrap aggregating. In the example case from the excerpt in table 6.3.1, this results in the network relationship described in table 6.3.2.

Table 6.3.2 Network Relationship: Bootstrap Aggregating/Simple Average

ϕ_1	ϕ_2	ϕ_3	ϕ_4	ϕ_5	ϕ_6	ϕ_7	ϕ_8	ϕ_9	ϕ_{10}	c
0.03	-1.00	0.00	0.19	-0.12	-0.16	0.14	-0.07	0.08	0.00	44.84

Next, the network relationship for each asset was computed identically to the description in algorithm 14 as well as the preceding Artificial Market Validation. Once both aggregation techniques were performed for all assets in the dataset, each assets' coefficients for all regressors were compared via cosine similarity. The nearest neighbour analysis took place on the resulting similarity matrix.

Inter-dependence Distributions

The primary concern sensor processing was implemented to address was that the sensor creation layer has the potential to overstate importance of over-represented factors. Since the factors are not directly observable the regressor asset allocations in each target asset's network relationship are reviewed.

Figure 6.3.6 displays the inter-asset dependencies in the simple average version for the example asset, ExxonMobil (XOM). Note that the greatest inter-asset association from the example asset is less than 2.5%. Furthermore, the distribution of asset dependencies was widespread. No group of assets was notably important to the price

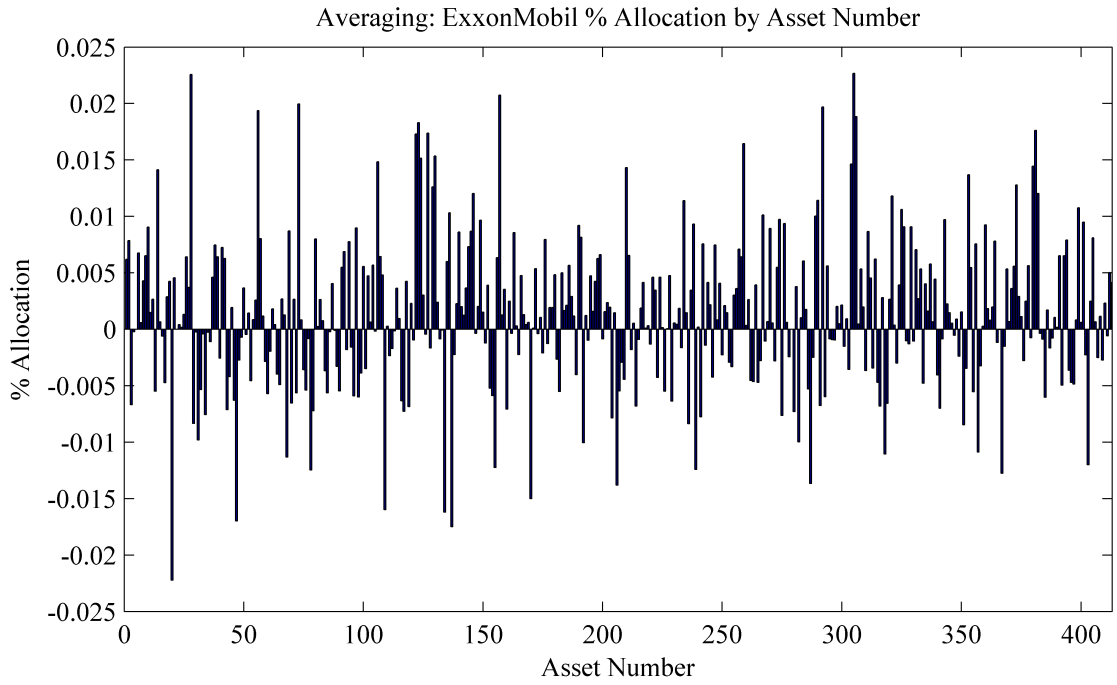


Figure 6.3.6: Sensor Processing: Simple Average XOM Network Relationship

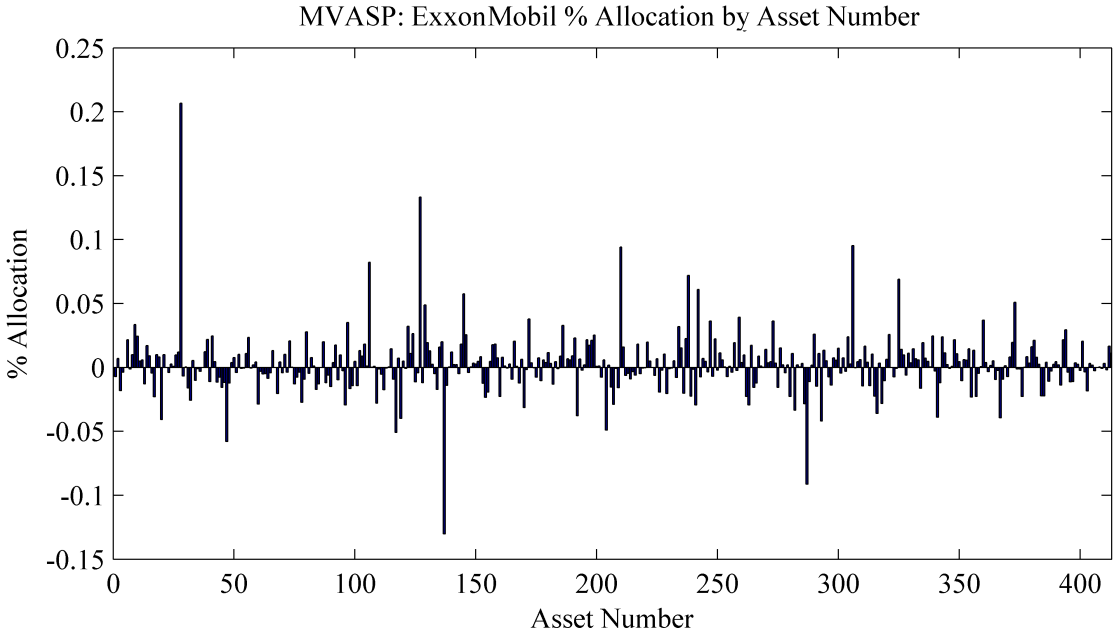


Figure 6.3.7: Sensor Processing: MVA Sensor Processed XOM Network Relationship

level of XOM. In general terms, it was expected for network relationships to contain a small set of assets which exhibited greater interactions representing the target asset's market sector. This was anticipated because of our previous assumption that assets in similar market sectors are most impacted by the same set of abstract pricing factors.

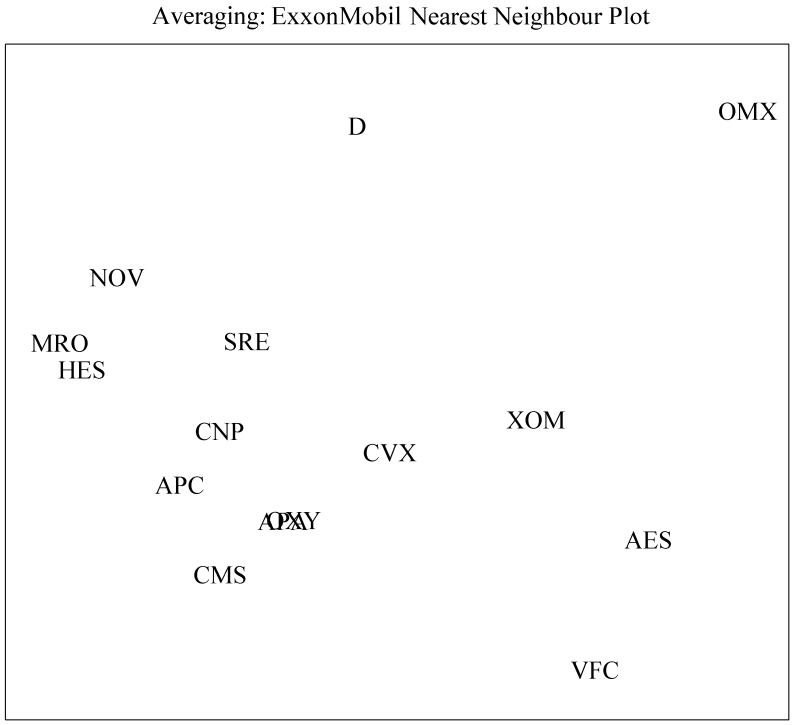
The averaging approach to processing was incapable of attributing significantly above average weights to a small number of assets. This is because each regressor asset's weight was limited by the number of times it was randomly selected to be a regressor. Contrastingly, the intermediate weighting of sensors during mean-variance sensor processing (MVASP) lessened the restriction on the size of a regressors' weight.

The inter-asset dependencies in figure 6.3.7 show that inter-asset relationships among sensor processed models aligned more closely with expectations. The inter-asset relationships were dominated by a small subset of assets while the majority of other regressors remained comparatively small. The maximum allocation to a single asset using the technique was over 9 times greater than the maximum allocation when averaging.

Multi-dimensional Scaling

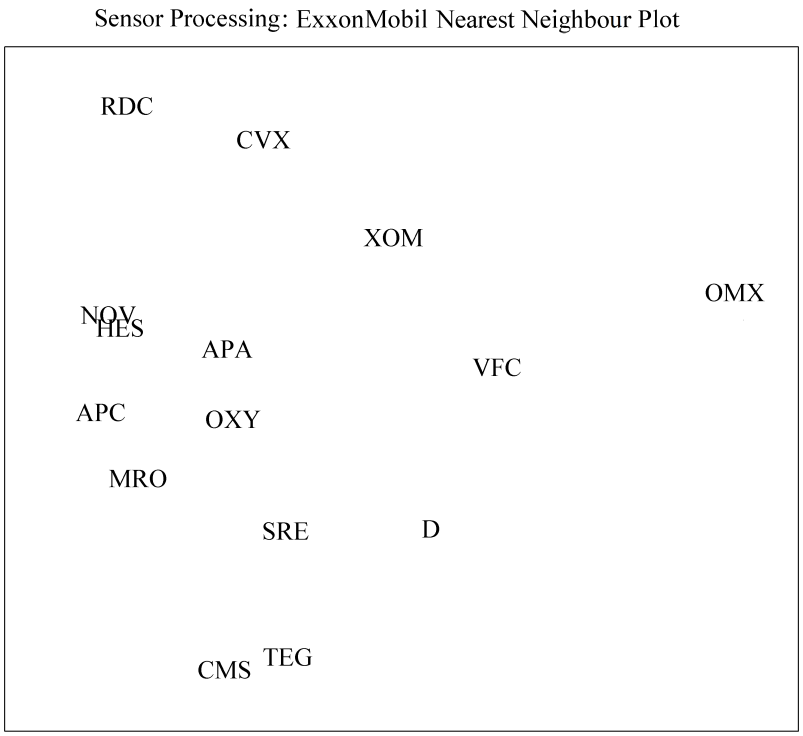
Simple average and MVASP network relationships were constructed for all 413 assets. To find an asset's nearest neighbours these models were first converted into a similarity matrix. Cosine similarity among assets' aggregate model regressor weights was used as the measure of similarity. In other words, each regressor became a dimension in 413D space. This allowed the network relationships to be represented as points in vector space. Cosine similarity of the points resulted with a 413 by 413 matrix for each approach.

Once a similarity matrix was found, any asset's M nearest neighbours were identified by locating the M shortest distances on its row of the matrix. It was expected that the MVASP approach would produce more intuitive results than the simple average. Through application of multi-dimensional scaling the distances computed in 413 dimensional space were projected on a 2 dimensional plane in figures 6.3.8 and 6.3.9. It was not possible to view large sections of the distance map without extensive distortion due to the large reduction in dimensionality. For this reason Exxon Mobil (XOM) was plotted with its 15 nearest neighbours only.



Ticker	Full Name	Industry
XOM	ExxonMobil Corporation	Oil & Gas
APA	Apache Corporation	Oil & Gas
CVX	Chevron Corporation	Oil & Gas
OXY	Occidental Petroleum Corporation	Oil & Gas
VFC	V.F. Corporation	Textile
AES	The AES Corporation	Electric Utilities
OMX	OfficeMax Incorporated	Retail
SRE	Sempra Energy	Gas Utilities
D	Dominion Resources, Inc.	Electric Utilities
APC	Anadarko Petroleum Corporation	Oil & Gas
CNP	CenterPoint Energy, Inc.	Diverse Utilities
CMS	CMS Energy Corp.	Electric Utilities
MRO	Marathon Oil Corporation	Oil & Gas
HES	Hess Corporation	Oil & Gas
NOV	National Oilwell Varco, Inc	Oil & Gas

Figure 6.3.8: Sensor Processing: Nearest Neighbour Averaging Exxon Mobil



Ticker	Full Name	Industry
XOM	Exxon Mobil Corporation	Oil & Gas
CVX	Chevron Corporation	Oil & Gas
APA	Apache Corporation	Oil & Gas
OXY	Occidental Petroleum Corporation	Oil & Gas
MRO	Marathon Oil Corporation	Oil & Gas
HES	Hess Corporation	Oil & Gas
SRE	Sempra Energy	Gas Utilities
APC	Anadarko Petroleum Corporation	Oil & Gas
NOV	National Oilwell Varco, Inc	Oil & Gas
CMS	CMS Energy Corp.	Electric Utilities
RDC	Rowan Companies plc	Oil & Gas
VFC	V.F. Corporation	Textile
D	Dominion Resources, Inc.	Electric Utilities
OMX	OfficeMax Incorporated	Retail
TEG	Integrys Energy Group, Inc.	Diverse Utilities

Figure 6.3.9: Sensor Processing: Nearest Neighbour MVASP Exxon Mobil

Results

The tables located below each of the plots are descending order lists of Exxon Mobil's most closely associated assets. As expected, the top few positions using the averaging technique were dominated by other oil and gas corporations. Some unexpected results were the presence of V.F. Corp (VFC), a textile manufacturer, and the OfficeMax (OMX) office supplies company. While its place on the list seemed high given the numerous other companies more similarly structured, V.F. Corp did contain significant interest in oil derivatives which explained the link. Excluding additional oil and gas corporations, the remaining places were filled by various utility companies. Given the overlap between big oil and utilities the association seemed plausible.

In summary, with the exception of 1 unrelated firm, the results did not explicitly violate intuitions. In order for the MVASP method to demonstrate improvement the asset's nearest neighbour order needed to more accurately reflect market sector expectations. In other words, it needed to rank a larger number of oil & gas corporations higher on the nearest neighbour list to be representative of an improvement. Another avenue for improvement was to rank OMX and VFC lower on the list.

Reviewing the the sensor processing approach in figure 6.3.9 a greater number of oil and gas corporations clustered at the top of the nearest neighbour list. Additionally, both of the questionable companies, OMX and VFC, were ranked lower. Given both results, it was clear that the MVASP approach violated expectations less than averaging. This further supported its use when constructing network relationships for each asset.

An unexpected negative cointegrating relation emerged between XOM and asset numbers 137 and 287 which correspond to NTRS or Northern Trust Corporation and ECL or Ecolabs Inc. respectively. Strong negative cointegrating relations suggest the companies returns are exactly opposite in responses to market factors. Considering NTRS was a financial holding company with $14 billion in assets under management while Ecolabs supplied cleaning products, the source of these relationships was not clear.

On further review, it was found that Exxon Mobil Corp. sued Ecolab Inc. in 2004 for trademark infringement [36]. A lawsuit is a example of when speculator's swinging expectations could cause a negative cointegration relationship between companies. The relationship would not have been discovered through use of

correlation. Over the sampled time period XOM and ECL returns were positively correlated with a value of 0.39. This provides an example of structure that associative models of financial markets can detect where traditional methods can not.

The negative relationship between Exxon Mobil and Northern Trust Corporation was less clear. In March of 2004 Exxon Mobil announced hiring Northern Trust Global Investments, a subsidiary of Northern Trust Corporation, to manage its defined contribution plan in excess of $\$2.5$ Billion. Although this would have had a direct impact on NTRS it is unclear how the relationship became negative.

6.4 Discussion

The sensor processing chapter began with and explanation of the primary shortcoming associated with the network model as applied in the sensor creation and sensor evaluation chapters. The foremost concern was the model's inability to attribute inter-asset dependencies greater than the number of times they were randomly selected for association in the sensors. This has the potential to skew asset's modelled dependencies. The source of the limitation was founded in the simple average approach to recombining sensor data. To resolve the issue the sensors were post-processed in the sensor evaluation layer to allow entry in mean-variance analysis. The application of mean-variance analysis enabled variably weighted recombination of sensors which mitigated the issue.

An example asset was observed with simple averaging and mean-variance analysis sensor processing to demonstrate the improvement. In both validations the MVA sensor processing was shown to improve network relationships of assets along selected performance criteria.

The entire process described in chapters $4-6$ leads to the creation of a network relationship for a single target asset. The network relationships were expected to contain information related to long-run market dynamics. The network model of the entire market is simply a collection of network relationships created where each asset in the market was the target asset for 1 network relationship.

In summary, the network model is a collection of cointegrating relations or network relationships. Each network relationship was created by holding an independent variable constant and regressing thousands of random subsets of available dependent variables. The similarity of resulting 'noisy' sensor models, or weak learners, was then evaluated. Based on this data a weighted combination of

sensors was found to improve the sensor's signal to noise ratio.

But we had a pretty diversified portfolio of businesses around the world and things tended to offset each other. But one or two years ago, we had a lot of things happening at the same time.

- Jim Cantalupo

7 Comprehensive Validations

7.1 Aims

To this point the 3 layer framework for creating network relationships, has been introduced. The network model of financial markets is simply the system of long-term relations found by collecting all of these models. There are 2 aims for the comprehensive validation chapter.

1. Demonstrate that the network model is a meaningful representation of financial markets

2. Demonstrate that a network model enables valuable and unique analyses

As repeatedly stated, the network model is expected to measure and describe the pricing patterns emergent from asset's shared dependencies on abstract market factors. These dependencies are immeasurable in financial markets which determines that the model can not be validated directly. For this reason, the network model must be validated by evaluating the accuracy of its composition and predictive capabilities.

The first validation extends the rudimentary algorithmic trading application from the intermediate layers of the framework. The extension is a consideration for market neutrality of the actively traded portfolio. This demonstrates that the model is a meaningful representation of markets via the accuracy its predictions, i.e. simulated returns, and introduces an unique approach to market hedging. It also shows the flexibility of the model by extracting expectations regarding future price movements and and market exposure from a single framework.

The second validation applies the network model to the task of measuring systemic risk in a financial market. The analysis is demonstrated to be a meaningful representation of financial markets if the implied relationships among market sectors do not violate expectations. It is valuable and unique given that it could provide insight into systemic connectivity among all market sectors from stable time data in aid of regulators, risk managers and researchers.

The selected validations were chosen to demonstrate the breadth of analysis available using an associative model. There are near boundless variations and adaptations of each analysis which could potentially yield interesting outputs. Given that each experiment is a validation to meet the outlined aims, minimal working examples are demonstrated in both cases. This leaves further experimentation along both directions of analysis available for future research.

7.2 Notation

Variable	Description	Sample Value
N	Number of assets in data set	100
L	Number of observations/period per asset	252
C	Matrix of daily close prices	Matrix [N by L]
ψ	Weight applied to a regressor in a final model	0.11
c	Constant term	7.84
ϵ	Error	Time-Series
s	Number of regressors per sensor	10
w	Number of sensors per target asset	5000

7.3 Method

There are 2 analyses provided. The first applies a market wide algorithmic trading approach on 2 financial markets for a multi-year evaluation period. The second validation uses the network model to review the implied systemic risk/connectivity profiles of various market sectors.

Both validations assume that the network model has been computed according to the description in chapter 3. An example of the output from a 10 asset network model in table notation is visible in table 7.3.1.

Table 7.3.1 Network Model: Network Relationship Table

ϕ_1	ϕ_2	ϕ_3	ϕ_4	ϕ_5	ϕ_6	ϕ_7	ϕ_8	ϕ_9	ϕ_{10}	c
-1.00	0.50	0.05	0.22	0.17	0.36	0.39	0.25	0.04	0.40	77.70
0.23	-1.00	0.12	0.36	0.09	0.35	0.47	0.27	0.38	0.19	23.79
0.11	0.15	-1.00	0.43	0.44	0.16	0.10	0.24	0.17	0.22	14.20
0.18	0.04	0.49	-1.00	0.20	0.24	0.29	0.02	0.45	0.32	84.96
0.11	0.17	0.41	0.45	-1.00	0.40	0.19	0.44	0.23	0.08	68.98
0.23	0.31	0.14	0.08	0.08	-1.00	0.46	0.33	0.02	0.19	63.16
0.03	0.11	0.17	0.04	0.09	0.29	-1.00	0.46	0.11	0.43	37.43
0.40	0.08	0.16	0.13	0.24	0.22	0.21	-1.00	0.48	0.37	73.01
0.16	0.26	0.17	0.19	0.26	0.31	0.14	0.14	-1.00	0.38	52.30
0.03	0.39	0.40	0.27	0.40	0.43	0.04	0.42	0.42	-1.00	96.49

Where each row in the table can be represented as a cointegrating relation and is expected to describe a long-run relationship in the market as in equation 7.1

$$0 = \psi_1 C_1 + \psi_2 C_2 + ... + \psi_N C_N + c + \epsilon \tag{7.1}$$

7.4 Validation

7.4.1 Algorithmic Trading Validation

Aims

The algorithmic trading validation shows a simulated trading strategy built on top of the network model's expectations. While many of the components for generating

positive expected returns have been implemented in intermediate steps of the framework, the comprehensive validation introduces considerations for portfolio market exposure which is made possible by the network model. The validation also includes transaction costs.

One aim of the chapter was to demonstrate that the network model is a meaningful representation of financial markets. the algorithmic trading validation meets this aim by utilizing out-of-sample predictions emergent from the network model to generate simulated returns. The second aim of the chapter was to demonstrate that the model enables unique and valuable analyses. In this validation we show value by introducing one potential approach to using the associative nature of the network model to manage a theoretically market neutral portfolio with positive expected return.

Method

There were 4 methodologies required to fully implement a theoretically market neutral algorithmic trading strategy. These methodologies are:

1. Defining expected market neutral portfolios
2. Defining trading signal with positive expected return
3. Portfolio initialization
4. Portfolio realignment

Expected Market Neutrality

Market neutral portfolios, as they pertain to the current validation, were defined as any portfolio expected to mean-revert to a value of 0.00 in the long-run. By this definition, holding the weights of any network relationship represented by a row in table 7.4.4 or similarly in the form of equation 7.1 meets this criteria. This is because every network relationship is a long-run cointegration relationship expected to mean-revert to a value of 0.00 in the long-run. Furthermore, any weighted combination of some group of the network model relationships is also expected to be market neutral for the same reason.

Positive Expected Return

Each expected positive return trading signal for an asset was derived from a single cointegration relationship in the network model. The relationship selected to

generate an asset's trading signal was the relationship created using the relevant asset as the target asset. For example, the positive expected return trading signal for asset number 3 would be created using row 3 of table 7.4.4. Continuing with the same example, the long-run cointegration relationship can be seen in equation 7.2

$$0 = 0.11C_1 + 0.15C_2 - 1.00C_3 + ... + 0.22C_{10} + 14.20 + \epsilon \tag{7.2}$$

Entering example values for each asset's current price yields equation 7.3

$$0 = 0.11(21.14) + 0.15(74.90) - 1.00(52.13) + ... + 0.22(44.57) + 14.20 + \epsilon \tag{7.3}$$

If the resulting relationship does not equal 0.00 when omitting the error term, ϵ, then the value of ϵ is set to satisfy the equation. During these times, the long-run relationship that the equation represents is thought to be temporarily out of alignment. In order for the long-run relationship to be satisfied without the error term, ϵ, certain information about future price movements is expected to be true. For example, if ϵ is large and positive then the value of asset 3 is likely to increase relative to the sum of all other assets in the relation.

In this way, the error term, ϵ, determines the direction of each asset's buy or sell trading signal. Repeating the process for each long-run cointegration relationship in the network model creates the buy and sell signals for each asset in the market. This process is identical to that used in the intermediate validations which utilized a simulated algorithmic trading strategy. The signals have no consideration for market exposure. For additional details see Chapter 2.2.2.

Estimated Transaction Costs

Adding transaction costs into the simulation create pressure for the portfolio's composition to change slowly. Note that adding market frictions actually obscures the view of how much market predictability the network model is able to identify. They are included for 3 reasons:

1. Realism
2. Equitable comparison with passively managed benchmarks
3. Demonstrate that predictability is meaningful

Adding realism to the validation was necessary as it contextualizes outputs with real world equivalents and benchmarks. While a comparison with real-world trading strategies is unfounded, including transaction costs notably reduces the discrepancy. The benchmark strategies used for comparisons in the results section are both passively managed. Excluding trading frictions may bias their comparison with the, actively managed, network portfolio. Furthermore, if the simulated predictability reaped from the network model is not valuable when accounting for minor frictions then the identified structure may be too insignificant for other research oriented analyses.

Market friction data was not available from considered sources [20, 38, 40] during the sample period. Historic data available from public sources did not include bid ask spread, quote, prices for durations exceeding the most recent few months. For this reason, transaction costs were estimated from available daily close prices from the sampled period and bid ask spreads for the month August 2012.

Historic market frictions were estimated by averaging values for each asset collected in August 2012. Using a relatively recent month's quote prices was inaccurate for estimating historic trading frictions due to the rapidly increasing trading volume in the last decade [39]. As trading volume for assets increases, bid ask spreads have decreased. This suggests that applying August 2012 market spreads to the S&P 500 data set spanning January 2005 to January 2014 would understate market frictions in oldest data and overstate frictions in the newest data. The approach was selected because it was the best available option, it is known to be decreasingly accurate as any simulated trading moves away from August 2012.

The expected cost of trading each asset was set to the 65th percentile of its spreads in August 2012. The minutely intra-day data were collected from the Reuter Eikon system[1]. Table 7.4.1 shows an excerpt of 10 assets and their estimated spreads.

Portfolio Initialization

The portfolio initialization task can be described as a single function maximization problem with multiple constraints. The goals of the comprehensive algorithmic trading validation were to maximize expected profitability subject to only trading theoretically market neutral portfolios.

[1]Thomson Reuters Eikon (UCL Student License)

Table 7.4.1 Excerpt of 65th Percentile August 2012 Spreads

Ticker Symbol	Estimated Transaction Cost/Share ($)
A	0.0107
AA	0.0101
AAPL	0.1180
ABC	0.0116
ABT	0.0111
ACAS	0.0099
ACE	0.0162
ADBE	0.0102
ADI	0.0111
ADM	0.0101

Creating a portfolio of equities from cash is a single-objective task. The only criteria a user must maximize is the expected profitability of the portfolio. The constraint of expected market neutrality was achieved by restricting portfolios to weighted combinations of network relationships/cointegration equations.

The portfolio initialization task falls into the category of combinatorial optimisation problems. A notable portion of these problems are solvable in polynomial time through application of linear programming[61] methods. Combinatorial problems falling outside this region can be approximated using numerous other metaheuristics which may be computationally expensive by comparison.

In this case, linear programming methods were unsuitable for 2 reasons.

1. Constraints are optional

2. Limited Multi-objective scalability

The description of market neutrality states that any portfolio of asset allocations found through a weighted combination of some subset of network relationships meets the criteria. It is not necessary to use all network relationships to find the allocations. This means that constraints are optional. Furthermore, linear programming was originally intended for single objective problems. There are numerous approaches to extending the concept to multi-objective problems[66] though there is no comprehensively applicable solution[2]. Given the constraint and scalability issues an approximation metaheurestic was selected for use.

The selected metaheuristic was a genetic algorithm or GA. A GA was chosen to implement the portfolio initialization and realignment tasks because it provides a

single approximation algorithm capable of performing both processes, i.e. single and multi-objective optimisations.

A single objective genetic algorithm as described in chapter 2 was applied to the problem to determine initial portfolio asset weights. In order to apply a genetic algorithm problem a fitness function was needed. The measure of fitness was the sum of least squares error between the solution candidate's portfolio allocations and the maximum expected positive return allocations.

The network model was computed where training data were all close prices in 2005. The sensor creation layer was parametrized with 8 regressors per sensor, s, and $1,000$ sensors per target asset, w. The sensor evaluation and processing layers allowed 20 sensors per 'sensor portfolio', δ.

The error levels in each asset's cointegration model which indicate expected future price movements for January 3, 2006 are visible in figure 7.4.1.

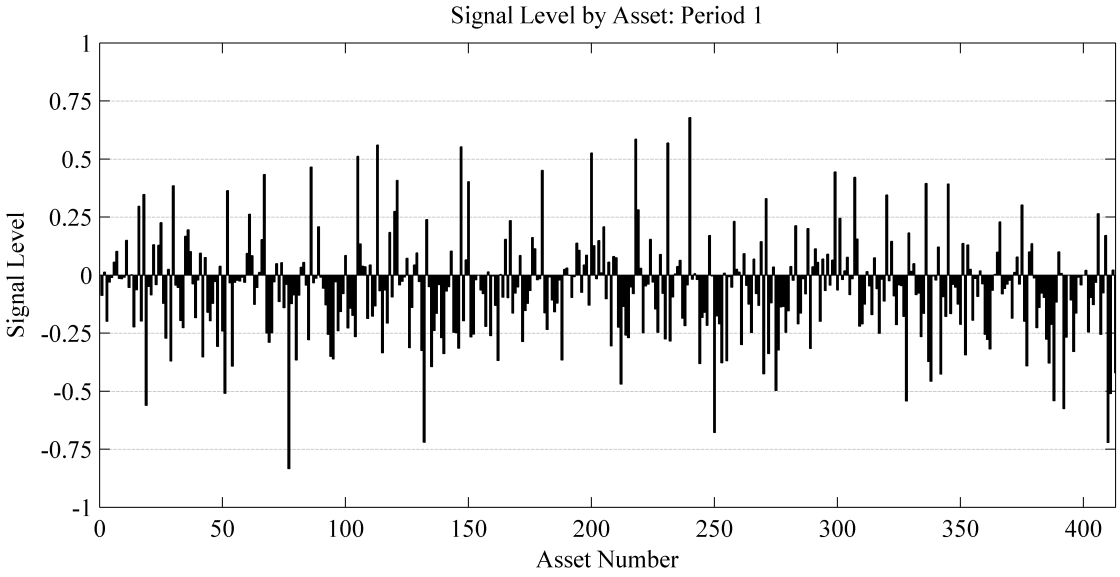


Figure 7.4.1: Multi-Asset: S&P 500 Initial Trading Signals

The remaining variables required to execute the genetic algorithm were, encoding, selection type, and recombination type. The encoding was chosen to be real-coded. Each individual in the population was composed of N weights between ± 2. The population size was then selected to an arbitrarily large value of $10,000$ solutions. To avoid the previously described over-exploitation concern, selection type was set to tournament with group size of 6. The crossover type was selected to be uniform after

trial runs evaluating uniform, one-point and two-point crossover on the metrics of convergence time and final solution quality after 300 generations.

The best solution found in the uniform crossover runs had a fitness value, or least squares error, of -8.1983. A histogram displaying the distribution of error by magnitude is shown in figure 7.4.2.

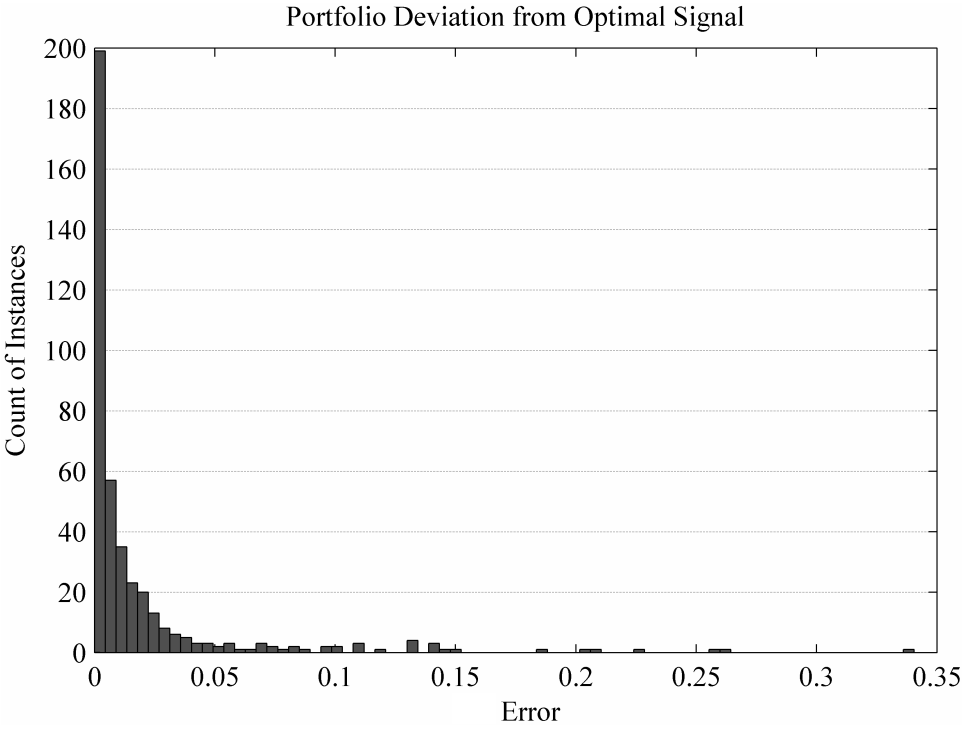


Figure 7.4.2: Multi-Asset: Trading Signal & Best Portfolio Initialization

In summary, the genetic algorithm solved for a combination portfolio weights which maximized expected returns subject to the portfolio being theoretically market neutral. This provides the starting point for the algorithmic trading simulation.

Portfolio Realignment

Portfolio realignment was performed at each period. At each time period the network model was recomputed. The updated network model was used to create expected returns and theoretically market neutral portfolios.

The method for portfolio initialization and portfolio realignment differ because initialization has a single objective to maximize expected return while the realignment must maximize expected return while attempting to minimize transaction costs. This made the problem multi-objective.

A multi-objective genetic algorithm was used to accommodate the the multi-objective portfolio realignment task. The selected multi-objective genetic algorithm was a parallel and GPU accelerated implementation of NSGA II created by the author [67] specifically for the task which reduced runtime 91 times compared to a single threaded implementation of NSGA II on the CPU.

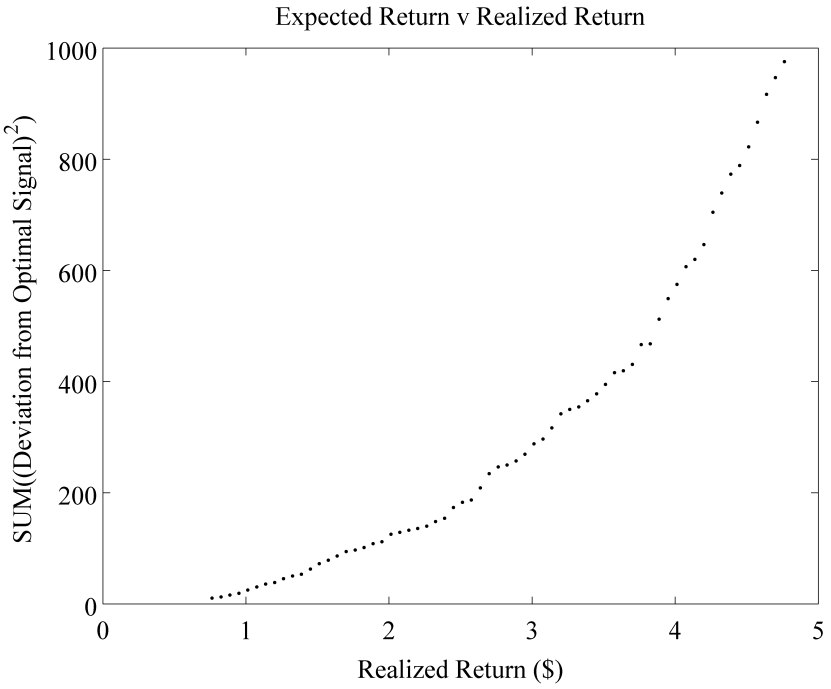


Figure 7.4.3: Multi-Asset: Sample Period Expected Return v Realized Return

Each point on the Pareto front represents a nondominated solution candidate in criteria space. The bottom left portion of the curve represents solution candidates which maximally align to the highest expected future return. Conversely, the top right portion of the curve represents solutions which close profitable positions inclusive of estimated transaction cost.

The solution candidate with realized returns nearest 0.00 was selected for every portfolio realignment. Selecting this candidate was intended to coerce certain emergent behaviours.

1. Prevent rapid entry or exit from positions
2. Balance transaction costs with realized returns
3. Dampen investment in assets until their trading signals are large

Preventing rapid entry and exit from the market was designed to reduce simulated transaction costs. If this was not considered the algorithm could consistently incur transaction costs in excess of realized returns. This would mean that the daily realignments could be a constant source of losses. Furthermore, selecting the solution candidate with realized returns nearest 0 incentivized trading positions with diminishing signals, i.e. historically profitable, to be cashed out and reinvested to better align the portfolio for future profitability.

A third behaviour expected to emerge from portfolio realignment was dampening the investment in assets until their trading signals were large. This has value because if a signal is growing period to period it is causing losses in any portion of investment currently in place. For example, if an asset A has a signal of -0.24 in period 8 which grows to -0.51 in period 20, an investment in A with between those periods with positive expected returns will generate loss. Dampening investment in assets until signals are large reduces losses by avoiding investment in assets while the momentum of their movement is unfavourable given the implied trading signal. A more direct approach to encourage this behaviour would be to employ a simple momentum based strategy which withheld investment until a leading and lagging moving average crossed. In other words, waiting until momentum began moving in the direction of profitability for the implied trading signal. This would be an interesting approach if the network model were being utilized for a practical algorithmic trading strategy but it adds an additional and unnecessary layer of convolution given that the selected portfolio realignment point was expected to provides loose guidance along these lines.

This analysis was performed on the 413 assets drawn from the S&P 500 index over the period Jan 3, 2005 to Jan 8, 2014. Given that portions of data were applied to previous analyses 2 precautions were taken to avoid biasing results due to data snooping. The first precaution applied was that the previously unused/unseen evaluation data set from October 2012 to January 2014 was independently reviewed. A second precaution was taken by applying the analysis to an entirely unseen data set.

The second data set selected were the constituents of the Australian Securities Exchange 100 (ASX 100). The ASX 100 is an index composed of the 100 largest market capitalization assets on the Australian Securities Exchange. The selected data were daily close prices ranging from May 1, 2006 to January 8, 2014. These assets were independent of all previous analyses.

The outlined methodology for the algorithmic trading validation was selected because a traded portfolio of long and short positions yields separable outcome data

which can readily provide insight into the network's performance. If a practical investor were to attempt to recreate the strategy, the portfolio could be hedged more efficiently through application of futures or options. Since the primary purpose of the validation is to verify the network model's expectations, the implemented experiment simultaneously verifies asset return and market neutrality expectations whereas more logical hedging choices would validate returns alone.

Results

Figure 7.4.4 shows the market movements of the S&P 500 and simulated cumulative return of 3 trading strategies. One of the trading strategies, designated 'Network Model', represents the network model trading strategy. The other two curves are naïve portfolio diversification strategies intended as benchmarks. The 2 methods selected as benchmarks were $1/N$ and mean-variance analysis portfolio optimisation.

The $1/N$ benchmark allocates a portfolio weight of $1/N$ to each available asset where N is the number of assets in the market. The strategy has no consideration for minimizing transaction costs if daily rebalancing were to take place. For this reason, allocations set at portfolio initialization were held static thereafter. This avoids unfairly decrementing the benchmark by guaranteeing no trading costs are incurred after portfolio initialization.

Standard mean-variance analysis was applied as a second benchmark as described in the chapter 2. All assets were made available to the analysis, short positions were allowed, and the risk-free rate was set to 0.00%. The portfolio's composition remained constant after initialization for the same reason as the $1/N$ strategy.

Comparisons in portfolio performance were made by comparing Sharpe ratio's among benchmarks. Table 7.4.2 compares portfolio performance over multiple time periods including the never seen, out-of-sample, dates.

Table 7.4.2 S&P 500 Portfolio Trading Sharpe Ratio Performance

Description	Dates	1/N	MVA	Network
All	2006/01-2014/01	0.423	0.759	1.343
Never Seen	2012/10-2014/01	2.076	0.771	0.786
Recession	2007/11-2009-01	-1.110	-1.085	2.089
Recovery	2009/01-2014/01	0.986	1.086	1.061

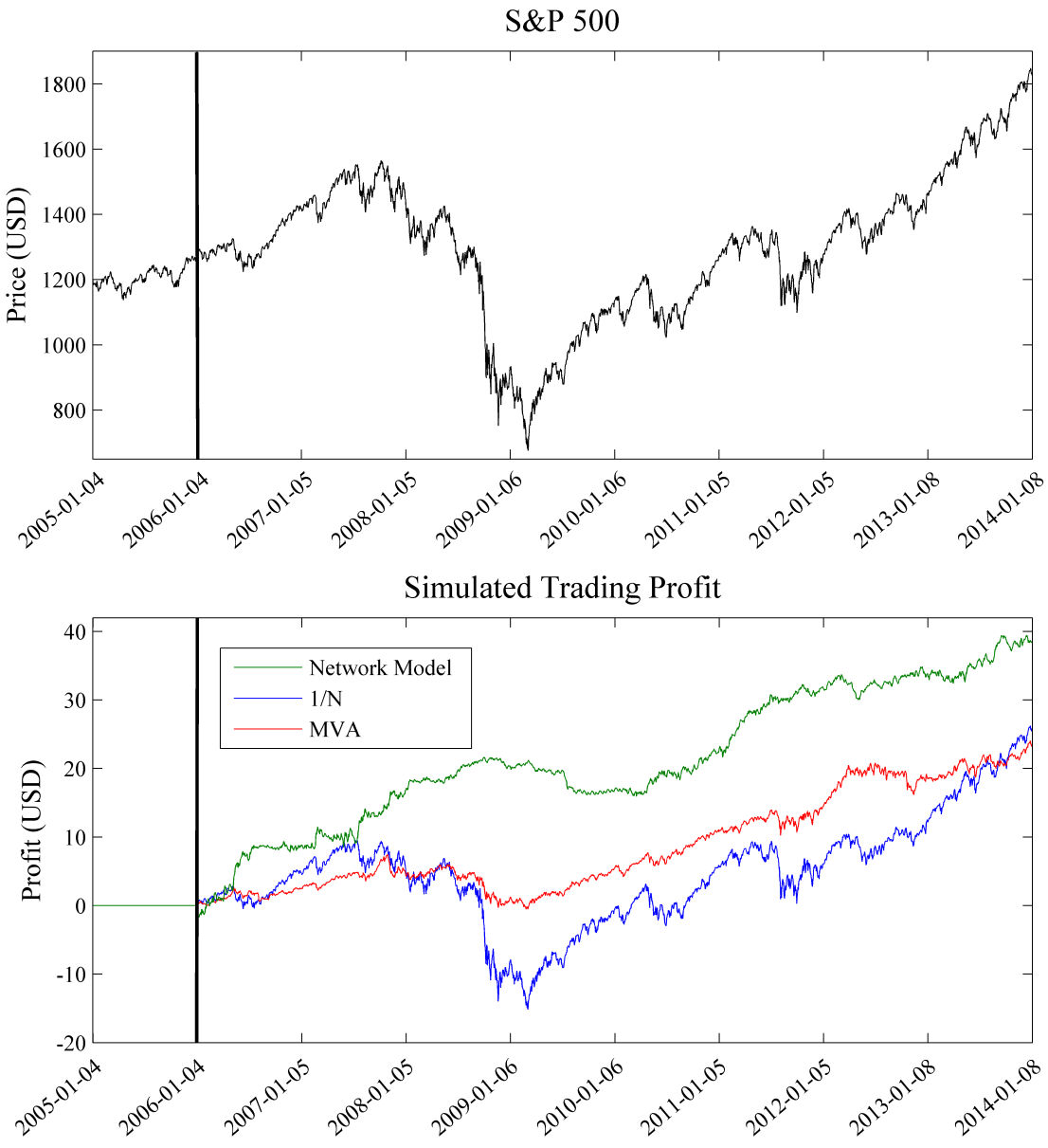


Figure 7.4.4: Portfolio Trading Performance v S&P 500

The values in table 7.4.2 show that all portfolios generated positive returns over the entire sample period. The network model's Sharpe notably exceeded those of the benchmark diversification strategies. The network model portfolio's excess performance relative to common diversification tools supports the model's validity. Excess performance supports the model's validity because the algorithmic trading strategy was based on the model's expectations. Therefore, the model is supported when the expectations tend to be accurate and generate consistent and positive returns.

During the 'Never Seen' period the 1/N benchmark significantly outperformed the MVA and the Network portfolios with values of 2.076, 0.771, and 0.786 respectively. The time period in question ranged from October 2012 to January 2014. Reviewing this section of the S&P 500 levels data in figure 7.4.4 shows that the market is characterized by rapid growth. The 1/N strategy containing long positions only captures this consistent market growth in the form of a high Sharpe ratio. Market exposure is not limited to downside losses. A portfolio that is significantly exposed to market movements will also participate in growth periods. That is what caused the high Sharpe ratio for the range of 1/N in the never seen period. This explanation is further supported by the strategy's under-performance during the recession period and lowest Sharpe ratio of 0.423 for the total 8 year evaluation period.

All listed time periods excluding the never seen section in table 7.4.2 include time periods which were applied in previous analyses and may be subject to data snooping. Therefore, these values are less conclusive than the never seen metrics. During the recession time period the network model algorithmic trading curve outperformed both MVA and 1/N benchmarks. The network model's expectations generated positive returns with a Sharpe of 2.089 while the benchmark curves sustained losses with Sharpe ratios less than -1.00. A speculative reason why recession performance was positive for the network model could be that the model fully immunized the portfolio from market exposure and the recession increased the likelihood of the long-run network relationships having large & profitable error terms. It is also possible that the analysis on this time range was biased by previous exposure to the data. To review this possibility, an identical analysis was performed on the Australian Securities Exchange (ASX). The results are located in figure 7.4.5 and table 7.4.3.

The S&P/ASX 100 trading validation also had 3 time ranges. In all cases these

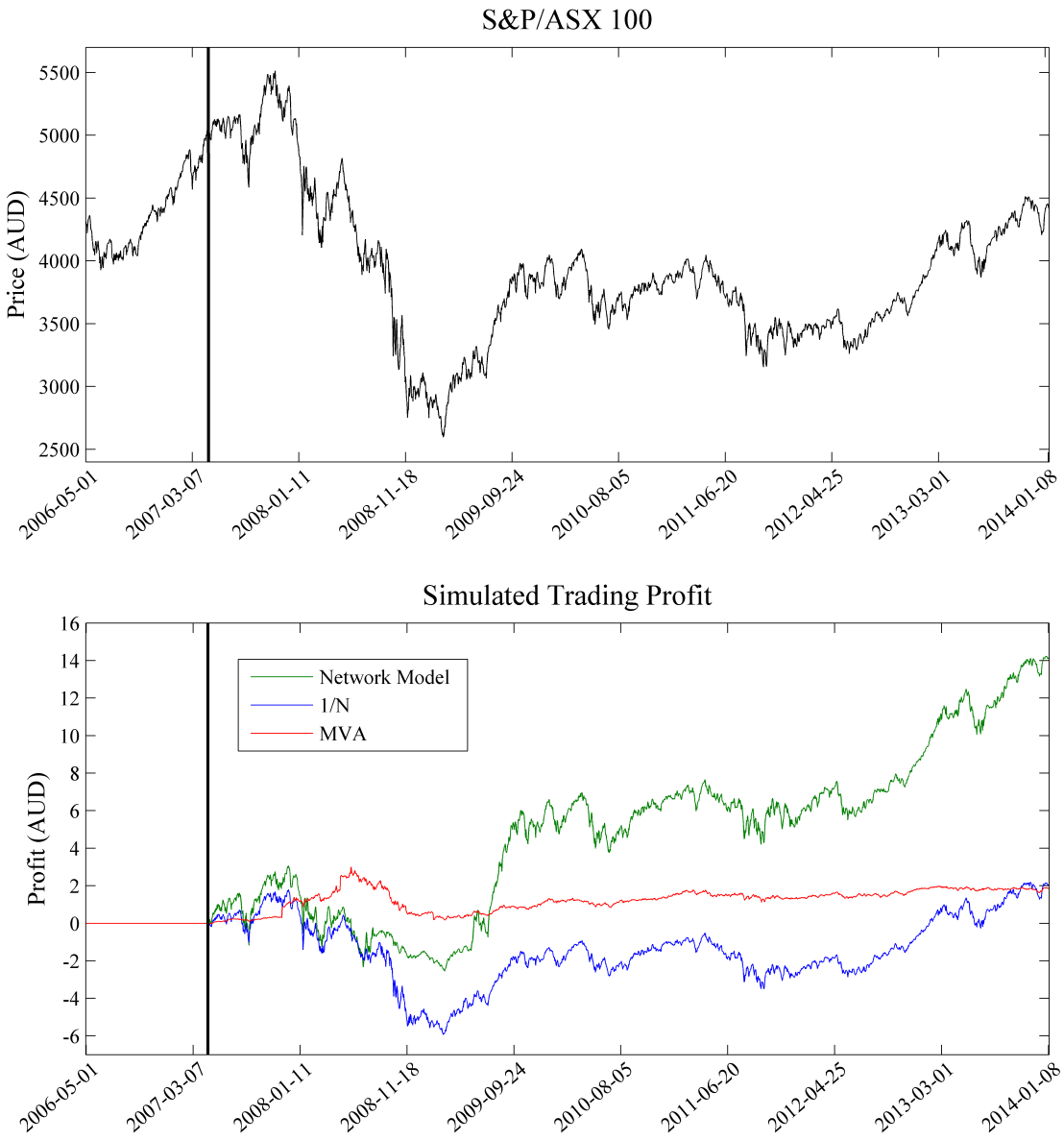


Figure 7.4.5: Portfolio Trading Performance v S&P/ASX 100

Table 7.4.3 S&P/ASX 100 Portfolio Trading Sharpe Ratio Performance

Description	Dates	1/N	MVA	Network
All	2007/04-2014/01	0.143	0.343	0.674
Recession	2007/11-2009-01	-1.820	-1.083	-1.166
Recovery	2009/01-2014/01	0.737	0.489	1.066

periods were out-of-sample and had never been seen in previous experiments. The network model outperformed 1/N and mean-variance analysis benchmarks over the entire sampled time range. It similarly outperformed these benchmarks during the recovery period. Both of these results support the network model's expectations relating to market neutrality and expected future price movements. During the recession period, the network model again outperformed the 1/N metric but performed comparably to the MVA portfolio. This outcome was not represented in the S&P 500 validation where the network model significantly outperformed the benchmarks. One possible explanation for the discrepancy could be that the comparatively small network of assets in the S&P/ASX 100 when compared to assets in the S&P 500 was insufficient to hedge the portfolio against market movements beyond the gains available through mean-variance analysis.

Given that the S&P/ASX 100 had no risk of data snooping the portfolio's behaviour was further reviewed. Figure 7.4.6 shows the source of portfolio returns split by long positions, short positions, and all.

It was expected that both long and short positions would be capable of generating positive expected returns. Figure 7.4.6 shows that short positions did not contribute to the portfolio's overall profitability. Short positions did not sustain significant losses over the experimental period. It is possible that predictions of asset price reductions were less accurate when compared with increases. It is also possible that composition of the network relationships made exploitation of long signals more readily accessible to the implemented portfolio realignment solver. While short positions did not contribute to the network's positive returns they did function as anticipated relating to market risk. Over the entire test period the network model produced a Sharpe ratio of 0.674. In comparison, the long portfolio produced a Sharpe of 0.599 which shows that short positions added value by reducing risk. The improvement is particularly significant since the short positions generated no positive returns. In other words, the risk mitigation associated with including the short positions improved the excess

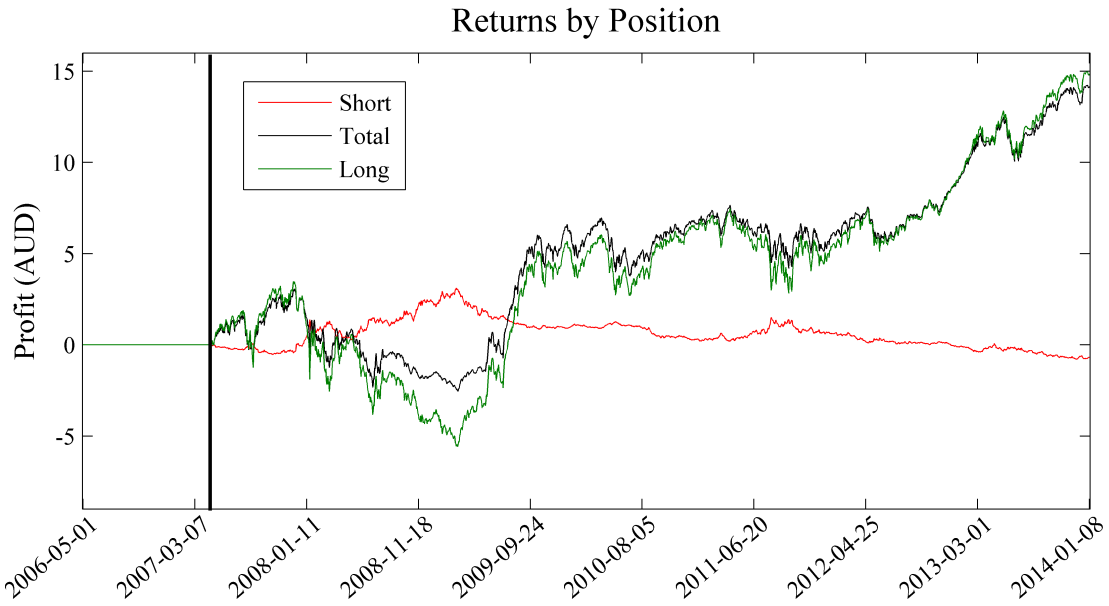


Figure 7.4.6: S&P/ASX 100 Returns by Long, Short, All

returns of the overall portfolio despite generating no returns. This clearly indicates that all associated value came from reduction in return variance. The recessionary period is the clearest example of the added benefit of the short positions. During this region the losses sustained by the long portfolio are visibly offset by the short portfolio.

The network's trading signal distribution for all assets in all time periods in figure 7.4.7 provides additional information relating to network model's inner workings. Comparing the cross section where frequency is 1000 and signal is ± 0.2 shows that the distribution of trading is skewed to buy recommendations. This provides another possible answer to the lack of profit in the short portfolio. If trading signals are skewed to buy recommendations then the network relationships believe that assets are more likely to be significantly under-priced than over-priced. In other words, it suggests that markets' estimates of asset prices are conservative.

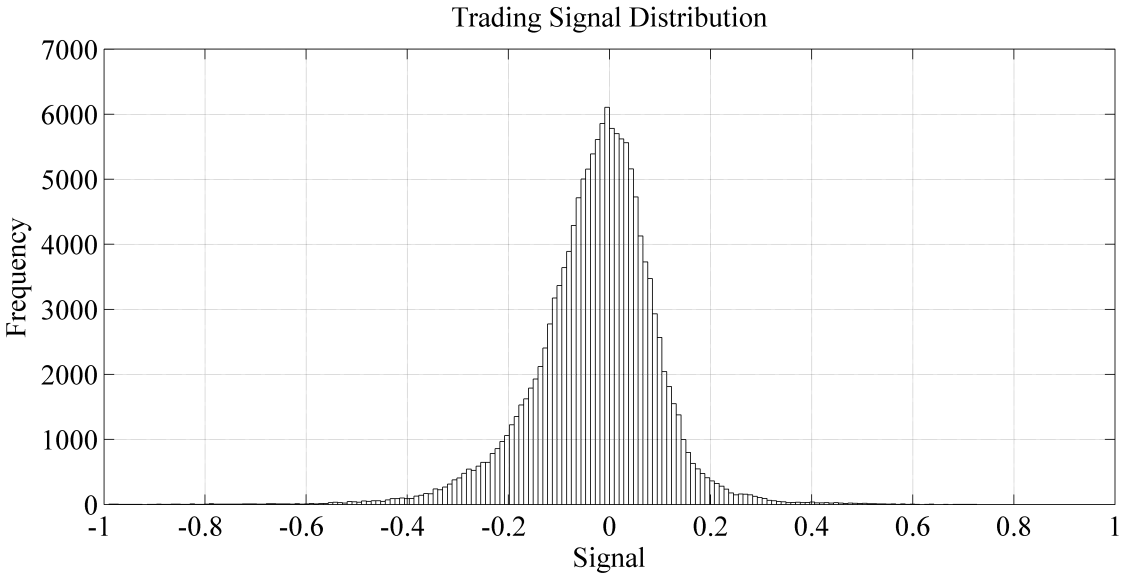


Figure 7.4.7: S&P/ASX 100 Trading Signal Distribution

7.4.2 Systemic Risk Validation

Aims

The systemic risk validation shows one analysis possible using the network model of financial markets. We aim to analyse the presence of systemic risk among market sectors in the New York Stock Exchange, NYSE. This validation is intended to address the second aim of the chapter. That aim is to demonstrate that the network model enables unique and valuable analyses.

Modelling a financial market as network of inter-asset associations creates a static representation with no new information being introduced. This makes the model ideal for assessing market dynamics relating to simulated price shocks applied to targeted assets. In the financial context this concept is known as systemic risk. Systemic risk is defined as the risk of the entire market incurring significant loss due to failure of a subset of that market's components[48].

We aim to use the network model to create a meaningful representation of systemic connectivity within a market at the sector resolution.

Method

The method of measuring systemic risk did not vary significantly from the approaches discussed in chapter 2. An initial shock is introduced into the market

through select assets, and a financial contagion function spreads the shock through the market. The impact of the shock can be observed by reviewing simulated prices post financial contagion. Market sector classifications for each asset were drawn from Google Finance[38]. The sector names, and abbreviations can be found in table 7.4.4.

Table 7.4.4 Market Sector Names and Abbreviations

Sector Name	**Abbreviation**
Basic Materials	B
Cyclical Consumer Goods & Services	CG
Non-Cyclical Consumer Goods & Services	NCG
Energy	E
Financials	F
Healthcare	H
Industrials	I
Technology	TEC
Telecommunication Services	TEL
Utilities	U

To better represent each market sector, the data set was not limited to the 413 assets from the S&P 500 used in other analyses. In this case, all available daily close common stock data from the NYSE was included subject to the aforementioned requirement of no sparsity over the sampled time period. The time period used for the systemic risk validation was Jan 1, 2013 to Jan 1, 2014.

Market sectors were defined as the market capitalization adjusted combination of all assets grouped by their designated name. Figure 7.4.8 illustrates the relative performance of each market sector over the training data. The period was marked by growth in every market sector.

Shocks were introduced into the simulation by applying a fixed percentage reduction to all assets in a single sector. In this experiment the selected shock size was 50%. For example, if the shocked sector was Technology, the initial shock was introduced by reducing the common stock price of all assets in the Technology sector by 50%. This reduced value was then held constant through time.

The financial contagion method was created from a minor adaptation of the network relationships. Presently, each network relationship in the network model is a cointegration equation expected to mean-revert to 0.00 in the long run. This concept

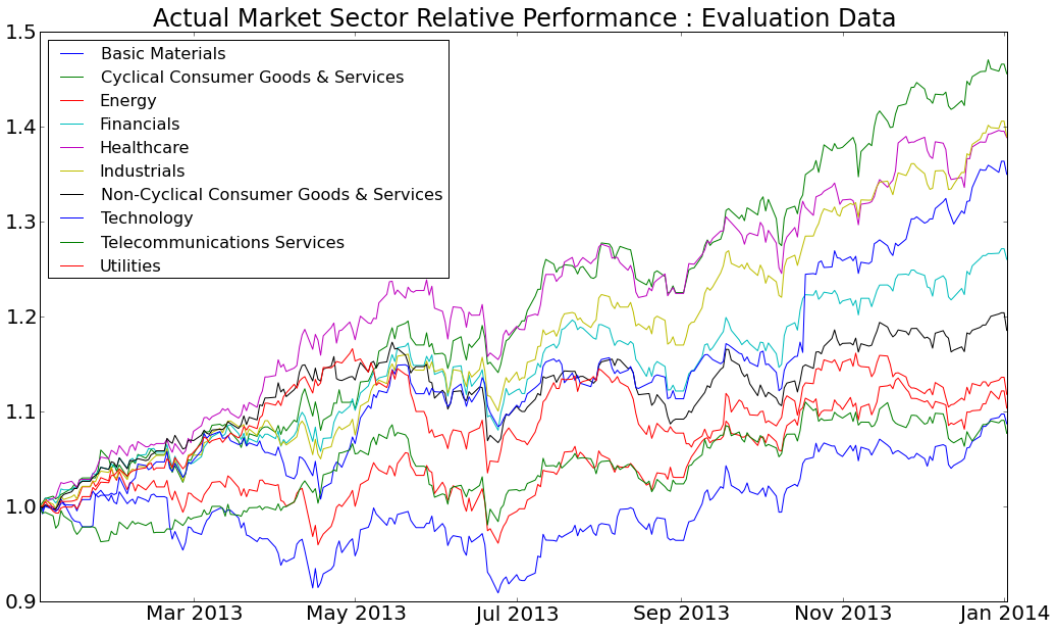


Figure 7.4.8: Market Sector Relative Performance

is re-shown below in equation 7.4.

$$0 = \psi_1 C_1 + \psi_2 C_2 + ... + \psi_N C_N + c + \epsilon \tag{7.4}$$

Though trivial manipulation, an asset's price can be estimated from known prices of all other assets. An example using the network relationship calibrated against asset 2 is shown in equation 7.5.

$$-\psi_2 C_2 = \psi_1 C_1 + ... + \psi_N C_N + c + \epsilon \tag{7.5}$$

Since ψ_i is always -1 for the i^{th} network relationship in the network model the $-\psi_i$ term can be dropped as in equation 7.6.

$$C_2 = \psi_1 C_1 + ... + \psi_N C_N + c + \epsilon \tag{7.6}$$

The processes of initial shock, financial contagion, and output collection are concisely described in algorithm 15.

Systemic risk was identified by simulating a pricing shock to a select sector and observing its propagation through the system. Performing this process for each market sector yielded a map of inter-sector exposures with an expected systemic impact and resilience for each.

Algorithm 15 Systemic Risk Validation: Shock Simulation

procedure SHOCK SENSORS

$N \leftarrow$ Number of assets in data set

$L \leftarrow$ Number of observations/periods per asset

$K \leftarrow$ Vector of asset's sectors of length N

$C \leftarrow$ Matrix of Daily Close Prices [N by L]

$t \leftarrow$ Number of simulated time periods

$V \leftarrow$ Shock simulation output prices [N by T]

$\kappa \leftarrow$ Shock size

top:

$ShockedSector = Energy \leftarrow$ Sector to be shocked

$\kappa = 0.5 \leftarrow$ 50% of original asset prices for shocked sector

for $i = 1 : N$ **do** $\leftarrow$ Loop to introduce inital shock on selected sector

if $K(i) == ShockedSector$ **then**

$V(i, 1) = C(i, 1) * (1 - \kappa)$

else

$V(i, 1) = C(i, 1)$

for $q = 2 : t$ **do** $\leftarrow$ Loop to spread financial contagion

for $i = 1 : N$ **do**

if $K(i) == ShockedSector$ **then**

$V(i, q) = V(i, q - 1) \leftarrow$ Hold shocked sector prices constant

else

$V(i, q) =$ *NetworkModel.predict*$(i, V(:, q - 1)) \leftarrow$ Spread shock

Where *NetworkModel.predict*$(i, V(:, q - 1))$ predicts the i^{th} asset's price from the i^{th} network relationship in the network model from the previous period's values of V. This is performed in accordance with equation 7.6.

In summary, the simulation implements a shock by reducing the value of all assets in the shocked sector. It then holds the shocked asset prices constant and uses the network model to predict price levels for all assets not contained in the shocked sector. Repeatedly predicting asset price levels from previous period values simulates the multi-period propagation of the initial shock throughout the market. This also prevents any new information from entering the system. Thus, all price movements in the simulation attempt to bring the asset prices into equilibrium with the network relationships. Final prices were reviewed after 140 iterations.

Results

Table 7.4.5 shows the total loss of market capitalization to each sector as a proportion of the initial shock's market cap. This accounts for variably sized sectors, and therefore variably sized shocks.

Table 7.4.5 Inter-Sector Exposures (Left = Shocked Sector, Top = Impacted Sector)

	B	CG	NCG	E	F	H	I	TEC	TEL	U
B	1.00	0.17	0.15	0.09	0.09	0.22	0.12	0.18	0.17	0.06
CG	1.03	1.00	0.49	0.31	0.10	0.41	0.32	0.12	0.18	0.16
NCG	0.67	0.20	1.00	0.12	0.12	0.20	0.20	0.15	0.14	0.06
E	0.95	0.27	0.44	1.00	0.12	0.38	0.37	0.22	0.02	0.11
F	1.97	0.60	0.72	0.51	1.00	0.75	0.59	0.61	0.44	0.20
H	0.80	0.29	0.33	0.30	0.26	1.00	0.31	0.26	0.07	0.09
I	0.99	0.31	0.40	0.28	0.24	0.34	1.00	0.29	0.31	0.22
TEC	0.91	0.32	0.45	0.24	0.20	0.28	0.33	1.00	-0.07	0.05
TEL	0.30	0.05	0.10	0.07	0.06	0.12	0.09	0.08	1.00	0.09
U	0.08	0.06	0.06	0.03	0.04	0.03	0.03	0.01	0.01	1.00

A visualization of the table as a network can be seen in figure 7.4.9. In figure 7.4.9 connections are directional and linkages are only shown where a constant minimum threshold of 0.50 was met.

It is apparent from table 7.4.5 that the network model views financial markets as inter-connected. The connections imply instability can arise from many market sectors and invariably has a magnified impact on the entire network. The inter-sector exposure table also uncovers some characteristics of the financials sector. Namely, financials were found to have the largest impact on external sectors while maintaining comparative resilience to outside shocks. As such, one would expect shocks to financials to spread rapidly to the remaining network. Other inter-sector exposures similarly align with expectations. For example, cyclical goods & services strongly impacts base materials with an expected ratio of 1.03 units of loss in the base materials sector per 1.00 units of loss to cyclical consumer goods and services. Similarly, sectors expected to be minimally connected, such as utilities, exhibited consistently small reactions to shocks in all other market sectors.

Figure 7.4.9 and table 7.4.5 indicate that the market is most exposed to shocks in financials. When setting a threshold on inter-sector exposures to 0.50 it was found

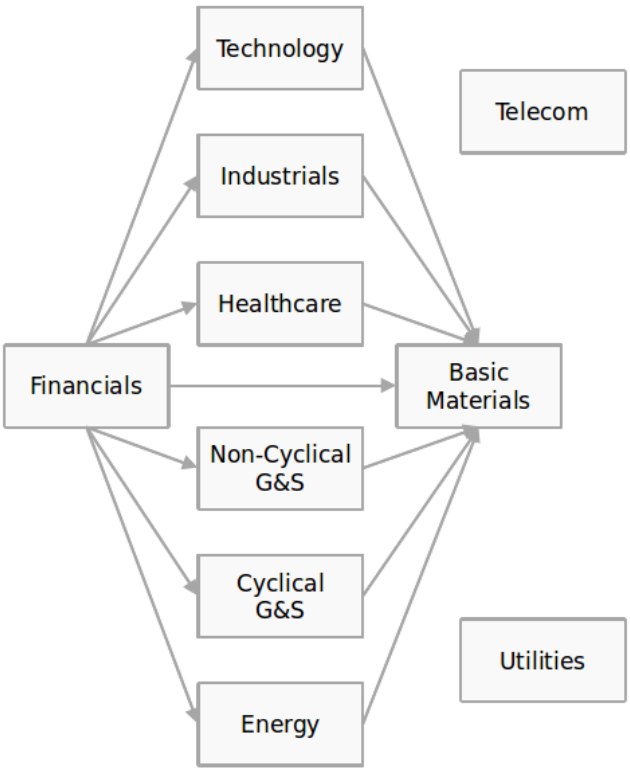


Figure 7.4.9: Inter-sector Systemic Exposure Network

that financials was outwardly linked to 7 of the 9 remaining sectors. In exact contrast, basic materials was impacted by the same 7 of 9 sectors. This suggests that financials is the most central 'hub' of systemic risk and basic materials is the most systemically dependent. A potential explanation for the strong inward connections to basic materials is that declines in numerous industries could rapidly reduce the need for raw inputs classified as basic materials.

The outcomes of this validation are two fold. We introduce a novel approach to modelling systemic risk within entire markets. By applying the model to New York Stock Exchange data we then estimate the direction and magnitude of inter-market-sector systemic connectivity.

7.5 Discussion

The chapter's aims were to demonstrate that the network model is a meaningful representation of markets and that it enables valuable and unique analyses. Both

validations provide comprehensive evidence in support of these points.

The algorithmic trading validation showed that the network model's expectations relating to portfolios of asset's market exposure and asset return were accurate. This was assessed relative to 2 benchmark portfolios implementing common techniques. It was shown that the network portfolio's Sharpe ratio performance was comparable to, or in excess of, the 2 benchmarks during multiple time ranges selected for interesting features, e.g. recession. The analysis was unique and valuable in that it utilized the only the network model to generate expectations relating to expected return and market neutrality. A single model capable of providing insight to sources of risk and return should be of interest in industry finance.

The systemic risk validation used the network model's cointegrating relations to model pricing shocks targeted to individual market sectors. The analysis used the network's connectivity data to simulate the spreading of shocks through the market from different origins. The result of the analysis was a inter-sector market exposure table. This information is valuable to both portfolio risk managers and policy makers. For example, the network determined that the Financials sector was the most systemically connected by a large degree. The network provides a clear and quantitative approach to modelling systemic risk among entire markets, as opposed to single sectors.

The global financial crisis in 2008 outlined the dangers of systemically connected corporate dependencies[57]. The global macroeconomic impact caused by the collapse of a small collection of institutions in the banking sector has created a surge of research efforts aiming to identify[4, 23, 77] and model[13, 35] systemic risk. Given the difficulty identifying systemically connected networks ex-ante, much of existing work has focused on the identified banking case where interbank lending agreements create a clear method for estimating systemic connectivity. Non-financial sectors provide a case with no equivalently obvious path forward.

It was shown that network models of entire financial markets provide one option for estimating presence of systemic risk both within and among financial sectors in a meaningful and data-driven way. Moreover, the systemic risk validation accurately determined that the Financials sector is highly connected from growth period data which implies it may be capable of detecting these features before they are highlighted by a market crash. This remains an avenue for future research.

The systemic risk validation met both aims of the comprehensive validation chapter. The validation demonstrated that the network model is a meaningful

representation of financial markets as it was able to differentiate among degrees of inter-market-sector systemic connectivity. It was further demonstrated that the network model enabled valuable and unique analysis of markets in that the most apparent trends in the output are of value to regulatory and decision making entities as well as having implications for portfolio risk management.

Both comprehensive validations provide evidence that the inter-asset relationships composing the network model are meaningful. Moreover, the applications demonstrates 2 areas where a network representation of markets can provide insight.

I am turned into a sort of machine for observing facts and grinding out conclusions.

- Charles Darwin

8

Conclusions & Future Research

This work is premised on a single concept that prices in financial markets may be guided by shared external factors. These factors may or may not be directly observable. The introduced network model attempts to identify and describe relationships among assets which emerge from mutually shared dependencies on external factors.

The model constructs a representation of financial markets using a 3 layer system.

1. Sensor Creation
2. Sensor Evaluation
3. Sensor Processing

The network model's output is a series of cointegration equations which encapsulate long-run trends in the form of inter-asset relationships. There was an output of 1 cointegration equation, or network relationship, per asset in the market.

The sensor creation layer assembled an arbitrarily large number of cointegration models containing random subsets of market assets as regressors. Each of these

sensors was expected to contain some small amount of information regarding the asset's shared factor dependencies.

Since these sensors were produced randomly it was known that the collection of information would be biased to over-represent external factors shared by many assets, and under-represent comparatively idiosyncratic information. The sensor evaluation and processing layers identify redundant information in sensors and recombine them to more accurately represent each asset's relationships within the market. The collection of resulting network relationships forms the basis of the network model. Each of the network relationships was expected to represent long-run market trends.

These relationships form a self contained information system ideal for simulations. Given that there was no way to directly evaluate the model's accuracy it was validated by observing its implications when applied to use cases. Throughout the thesis the model was applied to multiple unique challenges which consistently supported the model's validity.

Recalling the thesis statement:

> "We assert that assets in financial markets are mutually guided by arbitrarily defined external 'factors'. Furthermore, shared dependencies create emergent relationships among assets' price levels. We aim to leverage the inter-asset associations into a meaningful network representation of equities markets which enables unique and valuable analyses."

The first comprehensive application of the network model provided the most quantitative review of the validations. The model was used to predict future asset price movements in markets. This validation also used the network relationships to create theoretically market neutral portfolios with positive expected returns. The application confirmed that the algorithmic trading strategy applied from the network model's expectations either out-performed, or performed comparably to 2 standard benchmarks. Results were consistent across 2 markets, 8 years of data, and multiple market states.

The performance of the network model's portfolio strongly supports that inter-asset associations can create meaningful representation of financial markets. While it is not possible to claim that the source of the models' ability to make accurate predictions was drawn from identification of shared market factors, this explanation aligns with observed behaviour.

The second comprehensive application used the network model to evaluate the systemic connectivity, i.e. systemic risk, in a market. This demonstrated an intuitive methodology to modelling financial contagion when markets are shocked from any selected market sector.

Evaluating the implied systemic risk in financial markets readily identified the 'Financials' sector as far more systemically connected than remaining selected sectors. This was uncovered using growth period data. The ability to determine systemic connectivity ex-ante is a task currently subject to much research[8]. The analysis makes a convincing case for including a shared price level movement based network representation in this field. Most significantly, the network is not limited to banks or a single market sector as is common for other approaches. This is taken as an example of a unique and valuable analysis which further supports the thesis statement. Going beyond the thesis statement, the experiments provided evidence of interesting market features throughout the analyses.

The distribution of positive expected return trading signals in the algorithmic trading validation hints that markets may be conservative when pricing assets. The trading signal distribution was skewed towards long positions. This concept was shown over the entire testing period. The trading signal distribution was divided approximately $50/50$ between long and short signals and small short signals were notably more frequent than small long signals. Conversely, large long signals were notably more common than large short signals. That is, when assets deviated significantly from long-run network relationships they tended to be under-priced rather than over-priced. This is additionally supported by under-performance of the portfolio of short positions relative to long. The distribution provides evidence that market participants are inherently conservative in their estimates of asset prices.

The systemic connectivity/risk validation provided multiple insights about markets. The network model's relationships clearly identified the 'Financials' sector as the most systemically connected with 7 outward connections and 0 incoming at selected threshold levels. A significant portion of research efforts addressing systemic risk has focused on intra-sector systemic risk, primarily in the banking institutions. One possible reason for this is that inter-bank lending agreements create a convenient 'exposure' matrix to source financial contagion computations. The network model is an associative framework which extends our ability to simulate financial contagion among sectors.

The analysis also isolated 'Basic Materials', 'Telecommunications', and 'Utilities', as

having interesting connectivity properties. 'Basic Materials' was modelled to have the most incoming systemic connectivity by a large margin. This could be expected as shocks to any sector may reduce demand for their required basic materials. 'Telecommunications' and 'Utilities' were both isolated from the market at the chosen threshold level. The lack of incoming connections was expected given that demands for both sectors' products are relatively inelastic.

Remaining market sectors acted as additional pass-throughs of systemic connectivity from 'Financials' to 'Basic Materials'. This is further supported by the ratio of losses in 'Financials' to basic 'Materials' of 1 to 1.97. In other words, 1 unit of market capitalization loss in 'Financials' was simulated to create a 1.97 unit loss to market capitalization of 'Basic Materials'. This is 0.94 greater than the next largest exposure among any sectors, i.e. nearly double. The additional pathways between 'Financials' and 'Basic Materials' makes the value plausible and demonstrates consistency within the simulation. This also demonstrates the systemic connectivity validation's ability to capture multiple levels of financial contagion.

8.1 Future Research

The future possibilities for extending this work are numerous and fall into the following categories.

1. The Network Model
2. Applications

8.1.1 The Network Model

The network model is a system of network relationships generated by the introduced 3 layer framework. Each of these layers represents an opportunity for further improving the viability of an associative network model for financial markets. The primary direction of future research involves converting the modelling framework to leverage concepts from information theory. The intended outcome would be to more completely model relationships.

Sensor Creation: Layer 1

The sensor creation layer produced a vast set of noisy learners, or sensors, by iteratively creating cointegration models. These relationships were among a single common target asset and small random subsets of remaining assets. The expectation was that each learner would identify small amounts of information pertaining to the market dynamics among the target asset and the random subset.

The proposed sensors were linear sub-networks among assets. Holding relationships linear led to straightforward intermediate validations and 'root cause' backtracking of emergent behaviour. It also limits a possible source of confounding factors, significantly reduces the model's computational time, and limits its ability to over-fit the training data. Given that the uncovered linear network relationships were shown identify meaningful structure in the market, the restrictions could be relaxed to allow more complex associations. The introduction of nonlinearity is only valuable if the resulting network model contains greater information than the linear equivalent. One identified option for capturing nonlinearity while preventing over-fitting is implementing small binary trees. Individual sensors would be expected to capture significantly less information but could identify relationships with any level of complexity.

Sensor Evaluation: Layer 2

The sensor evaluation layer is responsible for accepting the low information content sensors and evaluating unique and redundant information. The information the user is most interested in identifying and modelling is dependent on the network model's intended application. For this reason is no single best method to improving the evaluation method. 2 general areas for further review in the layer are:

1. Incorporate feedback from the final model
2. Replace covariance with more flexible metric

Incorporating feedback from the final model could open the possibility of incrementally improving the model by determining how evaluation layer impacts the end result. While the opportunity could enable more accurate models it would create additional computational requirements, necessitate that the intended application was quantitative, and may result in over-fitting.

Replacing covariance with a more comprehensive and/or flexible metric would be universally applicable. One of the areas for future research from the sensor creation layer was the addition of capturing nonlinearity. This can similarly be an area for research in sensor evaluation. For example, replacing covariance with mutual information could enable nonlinear structure among sensors information contents to be recognized. Furthermore, a non-symmetric measure such as transfer entropy could similarly achieve this and distinguish between directions of information flow.

Sensor Processing: Layer 3

The sensor processing layer takes the evaluation data from layer 2 and uses it to recombine sensors into a single model which more accurately reflects the inter-asset relationships. The sensor evaluation and processing layers are heavily connected which means that the ability to further improve the sensor processing layer is contingent on the sensor evaluation layer's output.

Continuing with the suggested future research potential of utilizing information theory metrics, such as mutual information and transfer entropy, a similar concept could be applied in the sensor processing layer. A sensor processing layer operating on some form of 'information content' matrix would necessarily implement a solver to maximize the information content of the final network relationship. As with the other future research avenues, this would significantly increase computational requirements.

8.1.2 Applications

There were 2 minimal working example applications of the network model presented in the comprehensive validations section. Both of these applications are areas subject to significant future research.

Algorithmic Trading

The algorithmic trading model, as implemented, was intended solely to validate as many types of expectations drawn from the network model as possible. This leaves numerous options for furthering the research by attempting to create a practically efficient strategy. The 2 types of expectations which were applicable related to estimating expected return and producing theoretically market neutral portfolios.

The primary shortcomings to applying the network model as an algorithmic trading strategy relate to theoretical market neutrality. The implemented validation ensured that held portfolio weights always aligned with some combination of network relationships. The result of this was a guarantee that the portfolio would remain theoretically market neutral according to the relationships identified in the network model. This was valuable as it created the most direct implementation of the expectations.

In practice, there are financial instruments which are less expensive, less risky, and more conducive to hedging market positions than shorting common stock. For example, both futures and options could be applied to hedge risk of a portfolio based on the network model. These instruments could similarly be applied to position a portfolio for positive expected returns. It is likely that a practically applicable algorithmic trading strategy implemented on the network model's risk expectations would utilize multiple financial instruments. Identifying valuable methods of generating real-world profitability from the network model remains an area for future research.

Expectations relating to future returns are also subject to additional research. The next area selected for review is determining if the model specification and training data output is indicative of time to mean reversion. The discussed validation assumed that large trading signals were representative of the most profitable positions. If time to mean reversion deviates significantly among network relationships it may be possible to improve estimates of changes in out-of-sample asset prices levels.

There is no end to potential research which could improve any algorithmic trading system. A few areas for improvement include accuracy of transaction costs, accounting for slippage, lot size requirements, cardinality constraints, and accounting for intra-day impacts. These are general areas for algorithmic trading research which may be partially manageable by refining the network model's design.

Systemic Risk

The systemic risk validation provides a clear path forward for future research. The demonstrated implementation simulated shocks to sectors of the market and observed the resulting overall loss of market capitalization. The simulation discovered notable disparity between the systemic connectivities among sectors. These values aligned with general expectations of market dynamics.

Additional research could focus on analysing how shocks are disseminated across the market. The next intended area of research is to measure the extent to which the 'Financials' sector acts as a hub of systemic risk by passing negative feedback from shocked sectors to remaining sectors. Moreover, to determine if artificially removing 'Financials' from the market improves the overall stability.

Another interesting avenue for future work could be to evaluate the long-term return variance of portfolios which span the spectrum of systemic connectivity. For this analysis it would not be necessary to group assets by sector. Reviewing the level of structure of the resulting distributions in comparison to equivalently constructed portfolios based on a covariance matrix may produce interesting results.

Other

The applications of the network model in finance are numerous and clear but the model need not be restricted to the financial domain. Multiple, non-finance complex systems could potentially be modelled using an adaptation of the network model.

An example of such a system could be traffic patterns. Time-series data for fixed points in road networks could be collected from existing cameras and sensors which provides inputs to the network model. Other examples include analysing/predicting features of energy grids or weather patterns. As when modelling financial markets, these cases may benefit from some of the nonlinearity considerations discussed in the previous section.

8.2 Summary

The thesis began with the hypothesis that assets' are at least partially dependent on abstract external factors. It was further hypothesized that shared dependencies on factors could result in associative pricing patterns among financial assets.

This hypothesis was evaluated by creating an network representation of markets which modelled assets as a closed system of inter-asset dependencies. The network model was comprised of 3 layers. Layer 1 created an ensemble of weak learners expected to contain small amounts of information relating to the hypothesized emergent asset pricing patterns. Layer 2 evaluated similarities in information content among the weak learners. Layer 3 combined the weak learners to model a single asset in the network by drawing from the broad spectrum of information suggested by the

second layer. Repeating the process for every asset in the market produced the collection of relationships expected to capture the anticipated pricing patterns.

The network model was validated by testing its implied expectations to 2 comprehensive validations. Those validations were, an algorithmic trading simulation, and a systemic risk simulation.

The algorithmic trading validation applied the network model's expectations to 2 tasks. These tasks were, generating positive expected returns, and theoretical market neutrality. The validation demonstrated that the actively managed network model portfolio generated excess returns relative to 2 benchmarks. Over the time period of April, 2007 to January, 2014 the network model's expectations for assets drawn from the S&P/ASX 100 produced a Sharpe ratio of 0.674. This value approximately doubled the nearest benchmark. Benchmark Sharpe ratios were 0.143 and 0.343 representing 1/N and mean-variance optimal portfolios respectively. These figures support the network model's validity and value.

The systemic risk validation utilized the network model to simulate shocks to select market sectors and evaluate the spread of financial contagion throughout the market. The validation successfully differentiated sectors by systemic connectivity levels and suggested some interesting market features. Most notable was the identification of the 'Financials' sector as most systemically influential and 'Basic Materials' as the most systemically dependent. Additionally, there was evidence that 'Financials' may function as a pass-through or hub of systemic risk which exacerbates the financial contagion from multiple market sectors.

Both comprehensive validations demonstrate that the network model is able to capture meaningful structure among assets in financial markets. It is not possible to conclusively state that the identified relationships among financial assets resulted from mutual dependencies on abstract factors. Despite this, the model constructed specifically to exploit this theorized structure was demonstrated to capture valuable information across multiple test environments and time-frames. This strongly supports the thesis statement and underlying theory of market dynamics.

References

[1] B. Abraham and J. Ledolter. *Introduction To Regression Modeling*. Duxbury applied series. Thomson Brooks/Cole, 2006. ISBN 9780534420758.

[2] Maria João Alves and João Clímaco. A review of interactive methods for multiobjective integer and mixed-integer programming. *European Journal of Operational Research*, 180(1):99 – 115, 2007. ISSN 0377-2217.

[3] K.P. Anagnostopoulos and G. Mamanis. The mean–variance cardinality constrained portfolio optimization problem: An experimental evaluation of five multiobjective evolutionary algorithms. *Expert Systems with Applications*, 38(11): 14208 – 14217, 2011. ISSN 0957-4174.

[4] Stefano Battiston, Michelangelo Puliga, Rahul Kaushik, Paolo Tasca, and Guido Caldarelli. Debtrank: Too central to fail? financial networks, the fed and systemic risk. *Sci. Rep.*, 2(541), 2012. doi: 10.1038/srep00541.

[5] Sadek Benhammada and Salim Chikhi. A semi-formal specification for a generic model of artificial stock markets. *Procedia Computer Science*, 1(1):1465 – 1474, 2010. ISSN 1877-0509.

[6] Harald A. Benink, José Luis Gordillo, Juan Pablo Pardo, and Christopher R. Stephens. Market efficiency and learning in an artificial stock market: A perspective from neo-austrian economics. *Journal of Empirical Finance*, 17(4):668 – 688, 2010. ISSN 0927-5398.

[7] J.D. Bermúdez, J.V. Segura, and E. Vercher. A multi-objective genetic algorithm for cardinality constrained fuzzy portfolio selection. *Fuzzy Sets and Systems*, 188 (1):16 – 26, 2012. ISSN 0165-0114.

[8] Dimitrios Bisias, Mark Flood, Andrew W. Lo, and Stavros Valavanis. A survey of systemic risk analytics. office of financial research u.s. department of treasury, january 2012.

[9] Oleg Bondarenko. Statistical arbitrage and securities prices. *Review of Financial Studies*, 16(3):875–919, 2003.

[10] J. Branke, B. Scheckenbach, M. Stein, K. Deb, and H. Schmeck. Portfolio optimization with an envelope-based multi-objective evolutionary algorithm. *European Journal of Operational Research*, 199(3):684 – 693, 2009. ISSN 0377-2217.

[11] W. Brock, J. Lakonishok, and B. LeBaron. Simple technical trading rules and the stochastic properties of stock returns. *Journal of Finance*, 47(5):1731–1764, 1992.

[12] Derek W. Bunn. Time series analysis & forecasting: Lectures. University Lecture, 2011.

[13] J. M. D. Canedo and S. M. Jaramillo. A network model of systemic risk: stress testing the banking system. *Intelligent Systems in Accounting, Finance and Management*, 16(1-2):87–110, 2009. doi: 10.1002/isaf.295.

[14] Javier Márquez Diez Canedo and Serafín Martínez Jaramillo. A network model of systemic risk: stress testing the banking system1. *Intelligent Systems in Accounting, Finance and Management*, 16(1-2):87–110, 2009. ISSN 1099-1174.

[15] Jennifer N. Carpenter and Anthony W. Lynch. Survivorship bias and attrition effects in measures of performance persistence. *Journal of Financial Economics*, 54(3):337 – 374, 1999. ISSN 0304-405X.

[16] Allen Carrion. Very fast money: High-frequency trading on the NASDAQ. *Journal of Financial Markets*, 16(4):680 – 711, 2013. ISSN 1386-4181.

[17] T.-J. Chang, Nigel Meade, J. E. Beasley, and Yazid M. Sharaiha. Heuristics for cardinality constrained portfolio optimisation. *Computers & OR*, 27(13): 1271–1302, 2000.

[18] Mao cheng Cai, Xiaotie Deng, and Zhongfei Li. Computation of arbitrage in frictional bond markets. *Theoretical Computer Science*, 363(3):248 – 256, 2006. ISSN 0304-3975.

[19] Tak chung Fu, Chi pang Chung, and Fu lai Chung. Adopting genetic algorithms for technical analysis and portfolio management. *Computers & Mathematics with Applications*, 66(10):1743 – 1757, 2013. ISSN 0898-1221.

[20] Thomson Reuters Corporation. Reuters Eikon version 4, http://www.training.thomsonreuters.com/eikon4/, Date of Access: August 16, 2014.

[21] Hossein Dastkhan, Naser Shams Gharneh, and HamidReza Golmakani. A linguistic-based portfolio selection model using weighted max–min operator and hybrid genetic algorithm. *Expert Systems with Applications*, 38(9):11735 – 11743, 2011. ISSN 0957-4174.

[22] Kalyanmoy Deb, Amrit Pratap, Sameer Agarwal, and T. Meyarivan. A fast and elitist multiobjective genetic algorithm: NSGA-II. *IEEE Transactions on Evolutionary Computation*, 6, 2002.

[23] Danilo Delpini, Stefano Battiston, Massimo Riccaboni, Giampaolo Gabbi, Fabio Pammolli, and Guido Caldarelli. Evolution of controllability in interbank networks. *Sci. Rep.*, 3(1626), 2013. doi: 10.1038/srep01626.

[24] Giovanni Dosi, Giorgio Fagiolo, Mauro Napoletano, and Andrea Roventini. Income distribution, credit and fiscal policies in an agent-based keynesian model. *Journal of Economic Dynamics and Control*, 37(8):1598 – 1625, 2013. ISSN 0165-1889.

[25] Robert F Engle and Clive W J Granger. Co-integration and error correction: Representation, estimation, and testing. *Econometrica*, 55(2):251–76, 1987.

[26] Cain Evans, Konstantinos Pappas, and Fatos Xhafa. Utilizing artificial neural networks and genetic algorithms to build an algo-trading model for intra-day foreign exchange speculation. *Mathematical and Computer Modelling*, 58(5–6): 1249 – 1266, 2013. ISSN 0895-7177.

[27] Jiali Fang, Ben Jacobsen, and Yafeng Qin. Predictability of the simple technical trading rules: An out-of-sample test. *Review of Financial Economics*, 2013. ISSN 1058-3300.

[28] David E. Goldberg. *Genetic Algorithms in Search, Optimization and Machine Learning*. Addison-Wesley Longman Publishing Co., Inc., New York, NY, USA, 1989.

[29] Sven S. Groth and Jan Muntermann. An intraday market risk management approach based on textual analysis. *Decision Support Systems*, 50(4):680 – 691, 2011. ISSN 0167-9236.

[30] Axel Groß-Klußmann and Nikolaus Hautsch. When machines read the news: Using automated text analytics to quantify high frequency news-implied market reactions. *Journal of Empirical Finance*, 18(2):321 – 340, 2011. ISSN 0927-5398.

[31] Björn Hagströmer and Lars Nordén. The diversity of high-frequency traders. *Journal of Financial Markets*, 16(4):741 – 770, 2013. ISSN 1386-4181.

[32] C Hammel and W.B Paul. Monte carlo simulations of a trader-based market model. *Physica A: Statistical Mechanics and its Applications*, 313(3–4):640 – 650, 2002. ISSN 0378-4371.

[33] John H. Holland. *Adaptation in Natural and Artificial Systems*. University of Michigan Press, Ann Arbor, MI, 1975. second edition, 1992.

[34] Yu-Chia Hsu, An-Pin Chen, and Jia-Haur Chang. An inter-market arbitrage trading system based on extended classifier systems. *Expert Systems with Applications*, 38(4):3784 – 3792, 2011. ISSN 0957-4174.

[35] X. Huang, I. Vodenska, S. Havlin, and H.E. Stanley. Cascading failures in bi-partite graphs: Model for systemic risk propagation. *Sci. Rep.*, 3(1219), 2013. doi: 10.1038/srep01219.

[36] Lynn Hughes. Exxon Mobil Corporation v. Ecolab Inc, http://dockets.justia.com/docket/texas/txsdce/4:2004cv04236/357002, Date of Access: September 22, 2013.

[37] Mark L. Humphery-Jenner. Optimal VWAP trading under noisy conditions. *Journal of Banking & Finance*, 35(9):2319 – 2329, 2011. ISSN 0378-4266.

[38] Google Inc. Google Finance, http://www.google.com/finance/, Date of Access: Aug, 16, 2014.

[39] Google Inc. NASDAQ Composite Interactive Chart, http://www.google.com/finance?q=indexnasdaq:.ixic, Date of Access: August 16, 2014.

[40] Yahoo! Inc. Yahoo Finance, http://finance.yahoo.com/, Date of Access: August, 16, 2014.

[41] K. Izumi, F. Toriumi, and H. Matsui. Evaluation of automated-trading strategies using an artificial market. *Neurocomputing*, 72(16–18):3469 – 3476, 2009. ISSN 0925-2312.

[42] Durbin J. and Watson G. S. Testing for serial correlation in least squares regression: I. *Biometrika*, 37(3):409 – 428, 1950.

[43] Durbin J. and Watson G. S. Testing for serial correlation in least squares regression: II. *Biometrika*, 38(1):159 – 177, 1950.

[44] Durbin J. and Watson G. S. The sharpe ratio. *The Journal of Portfolio Management*, 21(1):49 – 58, 1994.

[45] Carlos M. Jarque and Anil K. Bera. Efficient tests for normality, homoscedasticity and serial independence of regression residuals. *Economics Letters*, 6(3):255 – 259, 1980. ISSN 0165-1765.

[46] Soren Johansen. Estimation and hypothesis testing of cointegration vectors in gaussian vector autoregressive models. *Econometrica*, 59(6):1551–80, November 1991.

[47] P. Jorion. *Value at Risk, 3rd Ed. : The New Benchmark for Managing Financial Risk: The New Benchmark for Managing Financial Risk*. Mcgraw-hill, 2006. ISBN 9780071736923.

[48] George G. Kaufman and Kenneth E. Scott. What is systemic risk, and do bank regulators retard or contribute to it? *The Independent Review*, 7(3), 2003.

[49] Joshua Knowles and David Corne. Approximating the non-dominated front using the pareto archived evolution strategy. *Evolutionary Computation*, 8: 149–172, 1999.

[50] Roman Kozhan and Mark Salmon. The information content of a limit order book: The case of an FX market. *Journal of Financial Markets*, 15(1):1 – 28, 2012. ISSN 1386-4181.

[51] Ludmila I. Kuncheva and Christopher J. Whitaker. Measures of diversity in classifier ensembles and their relationship with the ensemble accuracy. *Mach. Learn.*, 51(2):181–207, May 2003. ISSN 0885-6125.

[52] Jun Li and Jiuping Xu. Multi-objective portfolio selection model with fuzzy random returns and a compromise approach-based genetic algorithm. *Information Sciences*, 220(0):507 – 521, 2013. ISSN 0020-0255.

[53] Chang-Chun Lin and Yi-Ting Liu. Genetic algorithms for portfolio selection problems with minimum transaction lots. *European Journal of Operational Research*, 185(1):393 – 404, 2008. ISSN 0377-2217.

[54] D. Lin, S. Wang, and H. Yan. A multiobjective genetic algorithm for portfolio selection. 2001.

[55] Ping-Chen Lin and Po-Chang Ko. Portfolio value-at-risk forecasting with ga-based extreme value theory. *Expert Systems with Applications*, 36(2, Part 1): 2503 – 2512, 2009. ISSN 0957-4174.

[56] Yong-Jun Liu and Wei-Guo Zhang. Fuzzy portfolio optimization model under real constraints. *Insurance: Mathematics and Economics*, 53(3):704 – 711, 2013. ISSN 0167-6687.

[57] F. A. Longstaff. The subprime credit crisis and contagion in financial markets. *Journal of Financial Economics*, 97(3):436 – 450, 2010. ISSN 0304-405X. The 2007-8 financial crisis: Lessons from corporate finance.

[58] Viktor Manahov and Robert Hudson. Herd behaviour experimental testing in laboratory artificial stock market settings. behavioural foundations of stylised facts of financial returns. *Physica A: Statistical Mechanics and its Applications*, 2013. ISSN 0378-4371.

[59] Katiuscia Mannaro, Michele Marchesi, and Alessio Setzu. Using an artificial financial market for assessing the impact of tobin-like transaction taxes. *Journal of Economic Behavior & Organization*, 67(2):445 – 462, 2008. ISSN 0167-2681.

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

[61] Jirí Matouek and Bernd Gärtner. *Understanding and Using Linear Programming (Universitext)*. Springer-Verlag New York, Inc., Secaucus, NJ, USA, 2006. ISBN 3540306978.

[62] K. Metaxiotis and K. Liagkouras. Multiobjective evolutionary algorithms for portfolio management: A comprehensive literature review. *Expert Systems with Applications*, 39(14):11685 – 11698, 2012. ISSN 0957-4174.

[63] Melanie Mitchell. *An Introduction to Genetic Algorithms*. MIT Press, Cambridge, MA, USA, 1998. ISBN 0262631857.

[64] Douglas C. Montgomery, Elizabeth A. Peck, and G. Geoffrey Vining. *Introduction to linear regression analysis*. Wiley series in probability and statistics. Wiley, New York, United States, 3. ed edition, 2001.

[65] Andre R. Neveu. Fiscal policy and business cycle characteristics in a heterogeneous agent macro model. *Journal of Economic Behavior & Organization*, 92(0):224 – 240, 2013. ISSN 0167-2681.

[66] Carla Oliveira and Carlos Henggeler Antunes. Multiple objective linear programming models with interval coefficients – an illustrated overview. *European Journal of Operational Research*, 181(3):1434 – 1463, 2007. ISSN 0377-2217.

[67] Rickard Nyman Oliver Rice, Robert Smith. Approximating nonlinear effects during portfolio construction. *Theory and Practice of Natural Computing - Second International Conference, TPNC 2013, Cáceres, Spain, December 3-5, 2013, Proceedings*, 2013.

[68] Moustapha Pemy. Optimal algorithms for trading large positions. *Automatica*, 48(7):1353 – 1358, 2012. ISSN 0005-1098.

[69] Thorsten Poddig and Heinz Rehkugler. A 'world' model of integrated financial markets using artificial neural networks. *Neurocomputing*, 10(3):251 – 273, 1996. ISSN 0925-2312.

[70] Karolina Safarzyńska, Roy Brouwer, and Marjan Hofkes. Evolutionary modelling of the macro-economic impacts of catastrophic flood events. *Ecological Economics*, 88(0):108 – 118, 2013. ISSN 0921-8009.

[71] Andrea Schaerf. Local search techniques for constrained portfolio selection problems. *Computational Economics*, 20(3):177–190, 2002. ISSN 0927-7099.

[72] Hamed Soleimani, Hamid Reza Golmakani, and Mohammad Hossein Salimi. Markowitz-based portfolio selection with minimum transaction lots, cardinality constraints and regarding sector capitalization using genetic algorithm. *Expert Systems with Applications*, 36(3, Part 1):5058 – 5063, 2009. ISSN 0957-4174.

[73] N. Srinivas and Kalyanmoy Deb. Multiobjective optimization using nondominated sorting in genetic algorithms. *Evolutionary Computation*, 2: 221–248, 1994.

[74] Martin Wagener, Dennis Kundisch, Ryan Riordan, Fethi Rabhi, Philipp Herrmann, and Christof Weinhardt. Price efficiency in futures and spot trading: The role of information technology. *Electronic Commerce Research and Applications*, 9(5):400 – 409, 2010. ISSN 1567-4223.

[75] Chia-Hsuan Yeh and Chun-Yi Yang. Examining the effectiveness of price limits in an artificial stock market. *Journal of Economic Dynamics and Control*, 34(10): 2089 – 2108, 2010. ISSN 0165-1889.

[76] Brent Zenobia, Charles Weber, and Tugrul Daim. Artificial markets: A review and assessment of a new venue for innovation research. *Technovation*, 29(5):338 – 350, 2009. ISSN 0166-4972.

[77] Zeyu Zheng, Boris Podobnik, Ling Feng, and Baowen Li. Changes in cross-correlations as an indicator for systemic risk. *Sci. Rep.*, 2(888), 2012. doi: 10.1038/srep00888.

[78] Aimin Zhou, Bo-Yang Qu, Hui Li, Shi-Zeng Zhao, Ponnuthurai Nagaratnam Suganthan, and Qingfu Zhang. Multiobjective evolutionary algorithms: A survey of the state of the art. *Swarm and Evolutionary Computation*, 1:32,49, 2011.

[79] Hanhong Zhu, Yi Wang, Kesheng Wang, and Yun Chen. Particle swarm optimization (pso) for the constrained portfolio optimization problem. *Expert Systems with Applications*, 38(8):10161 – 10169, 2011. ISSN 0957-4174.

[80] E. Zitzler and L. Thiele. Multiobjective evolutionary algorithms: A comparative case study and the strength pareto approach. *Evolutionary Computation*, 3(4): 257–271, 1999.

[81] E. Zitzler, M. Laumanns, and L. Thiele. SPEA2: Improving the Strength Pareto Evolutionary Algorithm. Technical Report 103, Eidgenössische Technische Hochschule Zürich (ETH), May 2001.