Machine Learning for Decision-Support in Distributed Networks

Makgopa Gareth Setati

A dissertation submitted to the Faculty of Engineering and the Build Environment, University of the Witwatersrand, Johannesburg, in fulfilment of the requirements for the degree of Master of Science.

Johannesburg, December 2005

DECLARATION

I declare that this thesis is my own, unaided work, except where otherwise acknowledged. It is being submitted for the degree of Master of Science at the University of the Witwatersrand, Johannesburg. It has not been submitted before for any degree or examination at any other university.

Signed this ___ day of ____________ 20__

____

Makgopa Gareth Setati

MAIN ABSTRACT

In this document, a paper is presented that reports on the optimisation of a system that assists in time series prediction. Daily closing prices of a stock are used as the time series under which the system is being optimised. Concepts of machine learning, Artificial Neural Networks, Genetic Algorithms, and Agent-Based Modeling are used as tools for this task. Neural networks serve as the prediction engine and genetic algorithms are used for optimisation tasks as well as the simulation of a multi-agent based trading environment. The simulated trading environment is used to ascertain and optimise the best data, in terms of quality, to use as inputs to the neural network. The results achieved were positive and a large portion of this work concentrates on the refinement of the predictive capability. From this study it is concluded that AI methods bring a sound scientific approach to time series prediction, regardless of the phenomena that is being predicted.

ACKNOWLEDGEMENTS

The author would like to thank several people for their involvement during the research period: Prof Tshilidzi Marwala for his supervision and encouragement, Brain Leke for his guidance with the toolboxes, as well as proofreading. Evan Hurwitz for his assistance in looking for the data. I would also like to thank Armscor and CSIR for the financial support towards this work.

To my late father and grandfather, Bernard Makwelebeta Setati and Gilbert Makgopa Jerry Moloisi, this is for you. To my mother, Abigail Matema Setati, who gave me the strength and support to continue during a difficult time in our lives.

CONTENTS

TABLE OF FIGURES

LIST OF TABLES

MAIN INTRODUCTION

Aim of the work. The aim of this work is to describe an Artificial Intelligence (AI) approach to the prediction of time series values. Due to its availability and accessibility, the time series of daily closing prices for a stock is used (i.e. as proof-of-concept) to illustrate how AI techniques can contribute to the attempt of predicting the next day's closing price.

The work performed in this study is aimed at contributing to the research area involved with *Machine Learning for Decision-Support in Distributed Networks.* To put this in an elaborative context: this research area concerns itself with how the techniques of machine learning can be applied in an internetworked infrastructure of computers, such that intelligent decisions are arrived at. This is a vast area of research and because of this, my contribution emphasises on some of the optimisation aspects involved with such systems. Hence the title of the paper: *Optimisation of a Multi-Agent Stock Prediction System.* I am compelled to mention that the word *optimisation* is used sparingly in this context. This is because the field of optimisation is also vast and can thus not be exhausted in a single study. The details of how and which aspects are optimised is the subject of the main paper.

The title of the paper is thus qualified as follows:

- A multi-agent system is conceptualised in the context of distributed networks of computers and,
- the stock prediction capability puts the machine learning techniques into perspective. It then follows that the decision-support will stem from the predictions being performed by the system.

The paper presented here is intended to be as stand-alone as possible and it is written in a publishable style or format. This obviously places a constraint on the amount of information that is included in the paper. However, appendices are included for those readers that may need extended details.

Approach. In this study, Artificial Neural Networks (ANN) and Genetic Algorithms (GA) are used for machine learning purposes. ANNs are used as predictive engines whereas GAs serve as an optimisation engine that enhances the predictive functions. The philosophy underlying this work is twofold: First it is believed that the information being taught to a ANN must be both the right amount, and as 'clean' as possible. A GA is used to optimise this data and empirical results are used ascertain this fact. Secondly, it is known within ANN literature that the ANN architecture must be such that the best possible results are facilitated. Hence a GA-borne ANN architecture is developed such that predictions are optimised with particular objectives in mind, in my case the objectives are Euclidean distance and the correct prediction of direction.

In writing this paper and its supporting appendices, I have avoided becoming too deeply involved in the mathematical and statistical foundation for ANN and GA based methods. Furthermore, my philosophy urged for the pursuit of methods that yield good performance in practice and hence eliminating the need to explore other subservient methods. As a consequence, the focus is placed on two-layer Perceptron neural networks with hyperbolic tangent units and linear outputs units. This is probably the most commonly used network architecture as it works quite well in many practical applications. However, besides the appendices, the reader is referred to more fundamental textbooks on ANNs for treatment of other types of neural networks.

Supporting software. All simulations are run using the mathematical software package MATLAB®:

- The NETLAB Toolbox, which contains a collection of MATLAB® functions that can be used as tools for design and simulation of neural networks.
- The GAOT Toolbox, which contains a collection of MATLAB® functions that can be used as tools for design and simulation of evolutionary machine learning.

A detailed description of the contents and use of the two packages can be found in the manuals by Ian Nabney and Christopher Houck et al. The manuals are titled: *NETLAB - Algorithms for Pattern Recognition* and *A Genetic Algorithm for Function Optimisation: A Matlab Implementation* respectively. Software and manuals can be downloaded from the internet at:
http://www.tech.plym.ac.uk/spmc/matlab/matlab_toolbox.html

Prerequisites. Primarily, this work addresses professionals in the following research fields: electrical engineering, computer science, actuarial science and related commerce areas. However, professionals in other research areas may find the work implicitly relevant either for self-study or as a guideline for implementation. As a minimum, the reader should know about matrix calculus, basic statistics, time series analysis, and basic optimisation techniques. It will also be a significant advantage if the reader is acquainted with the ANN, GA, and investment fields.

Outline of the paper. Section 1 introduces the paper by motivating and qualifying the work performed. Section 2 and 3 briefly outline the theoretical foundation of GAs and ANNs respectively. Section 4 outlines the methodology for designing and optimising a single component of the system. Results are discussed and interpreted in Section 5; and Section 6 concludes this study.

Optimization of a Multi-Agent Stock Prediction System

Makgopa Setati
School of Electrical and Information Engineering
University of the Witwatersrand
Private Bag 3, 2050
Johannesburg, South Africa
g.setati@ee.wits.ac.za *or* garethsetati@yahoo.com

Abstract

In this study we make use of a genetic algorithm (GA) and artificial neural networks (ANN) to perform a one-step prediction of the next day's closing price. A multi-agent framework is conceptualized so that work is concentrated on the optimization of a single agent's prediction engine. An optimal genome using a GA is arrived at, and the genome is used to generate an input data matrix of optimized technical indicators. The technical indicators are used to train an artificial neural network that performs a prediction. Prediction lags are experienced and three approaches from previous literature are explored in order to deal with the lag. The first approach involves using additional inputs into the neural network and the lag was reduced from 4 days to one day. The second approach proposes a method of measuring the predictability by using hit rates. The hit rates were calculated as a measure of how well the neural network predicts the next day's direction of change; a hit rate of 72% was achieved. However the method of computing this hit rate was criticized and hence a better computation technique is suggested. Lastly, a multi-objective GA approach is theoretically suggested, then modified and implemented as a method of addressing the lag by maximising the prediction of the next day direction of movement, as well as minimising the mean square error. A maximum of 77% and a minimum of 54% direction success were achieved. The study concludes by recommending a multi-objective approach that incorporates a timing error (lag) as well as an adaptation mechanism for the ANN inputs and architecture.

1 INTRODUCTION

Detecting and predicting trends of market stocks is a difficult task. This is ensured by the highly correlated economic, political, and psychosocial factors that affect the stock's fluctuations in price. To this effect, a substantial amount of research in the area of stock forecasting has been conducted. Currently, technical and fundamental analyses are the dominant forecasting methods in use. Technical analysis is a method of evaluating stocks by analysing the statistics generated by market activity, past prices, and volume. In other words technical analysis does not attempt to measure a stock's intrinsic value and/or a company's financial health; instead it provides a framework for studying investor behaviour by focusing only on price and volume patterns. Traders using this approach usually have short-term investment horizons and have access to only past prices and exchange data. On the other hand, fundamental analysis is a method of evaluating future prospects of stocks by analysing the financial statements of the share's company. Followers of this approach want as much data and information on 'metrics' like revenues, expenses, assets, liabilities, and so forth.

One of the pillars of current financial theory is the Efficient Market Hypotheses (EMH) [1]. EMH and the Random Walk Theory state that at any time, the price of a share fully captures all known information about the stock and that since all known information is used optimally by market participants, price fluctuations are random, because information occurs randomly. Thus, a stock price performs a "random walk" and is therefore impossible for an investor to outperform the market. In other words, markets are efficient in that opportunities for profit are discovered so quickly that they cease to be opportunities. To date, researchers and academics debate the validity of EMH and there has not been a consensus on the hypothesis. However, in [2] Tan and Yao used a rescaled range analysis and market efficiency testing [3] [4] to distinguish between a random series and a fractal series, irrespective of the statistical distribution of the underlying series. This is a robust statistical test for measuring the amount of noise in a system and is used to determine the persistence of trends and the average length of non-periodic cycles. The results obtained in their study implied that markets are not random walk and are not efficient enough therefore forecasting markets, is possible.

Traditionally, statistical regression models such as the autoregressive integrated moving average (ARIMA); popularly known as the Box-Jenkins [5] methodology, dominate time series prediction. A time series is an ordered sequence of observations made through time, $x(t), t = 0,1,...,n$ where a time series prediction is the prediction of future values for a specified horizon $h, \hat{x}(t+h), h > 0$ given historical data. Recent breakthroughs in computational intelligence and numerical algorithms have given rise to many methods in financial engineering. Artificial Neural Networks (ANN) are one such area and have thus gained popularity as a technique for forecasting stock prices. In addition, empirical results have shown that neural networks outperform linear regression [6][7] because stock markets are complex, nonlinear, dynamic and chaotic. This is because ANNs make few assumptions as opposed to normality assumptions usually found in statistical models – ANNs perform predictions after learning the underlying input/output relationships of the data [8]. From a statistician's point of view, neural networks are analogous to nonparametric, nonlinear regression models.

This paper describes the application of artificial intelligence techniques in performing a one-step prediction of a single stock. Genetic Algorithms (GA) and ANNs are conceptualised in a Multi-Agent framework that performs stock price forecasting. Firstly a GA is used to simulate a Multi-Agent stock trading population such that the best agent is selected and then equipped with an ANN that performs the prediction. Secondly a GA is used to optimize a set of variables that are used as inputs to a predictive ANN. These two approaches are used to test the hypotheses that the most profitable trader in a stock market will make the best one-step predictions. The best approach of the two was then used as a framework under which the prediction system will be based. Technical indicators and time series data are used as inputs to the ANN and GA is once again used to optimise the final architecture of the ANN. The time series data used was obtained from the closing prices of the South African Breweries (SAB) stocks from 31/12/1999 to 01/01/2004.

Predictions are performed, and various strategies for performance measurement and improvement of the results are presented. Section 2 and 3 briefly outline the basis behind GAs and ANNs respectively. For brevity within the paper, a more detailed theoretical outline of ANNs and GAs can be studied in appendix B and C respectively. Section 4 outlines the methodology for designing the agent of interest. Results are discussed and interpreted in Section 5; and Section 6 concludes this study.

2 GENETIC ALGORITHMS

In this section a brief outline of GAs is presented for the reader that is not interested with the in-depth theoretical dynamics. A detailed theoretical framework under which GAs operate can be studied in appendix C. The following books are suggested [9][10] for the readers who may also be interested in the underlying mathematical workings of GAs.

During the last four decades, interest in problem solving systems based on the principles of evolution and hereditary has grown: such systems maintain a family of potential solutions; they have a certain selection criteria based on the fitness of individuals, and some “genetic” operators. The common term in use, Evolution Programs (EP), for all evolution based systems. One such type of an EP is the GA. The structure of an evolution program is shown in Figure 1.

A GA is a probabilistic algorithm that maintains a population of individuals, $P(t) = \{x_1^t, ..., x_n^t\}$ for iteration t. Each individual represents a potential solution to the problem and is implemented as some data structure. Each individual solution x_i^t is evaluated to yield some measure of its “fitness”. Then a new population (iteration $t+1$) is formed by selecting the fittest individuals. Some members of the population undergo transformations by means of unary and higher order transformations. The former creates new individuals by a small alteration (mutation) in a single individual whereas the latter creates new individuals by breeding (crossover) parts from several individuals. After some specified generations (iterations), the program converges and it is hoped that the fittest individual from that population represents a near-optimum solution.

```
PROCEDURE EVOLUTION PROGRAM
begin
    t = 0
    initialise P(t)
    evaluate P(t)
    while (not termination-condition)
        t = t + 1
        select P(t) from P(t - 1)
        alter P(t)
        evaluate P(t)
    end
end
```

Figure 1. The Structure of an Evolution Program [9]

GAs are extensively used in optimisation problems whose objective functions are difficult and do not possess desirable properties such as continuity, differentiability, satisfaction of the Lipschitz Condition, etc. [9].

3 ARTIFICIAL NEURAL NETWORKS

As in Section 2, only a short overview of ANNs is presented in this section. Readers may study further details on ANNS in appendix B and the excellent book by Bishop [11].

ANNs are mathematical models inspired by biological (brain) models. The models consist of interconnected elements known as neurons that are arranged in layers. An ANN receives signals through the input layer and these signals are propagated and transformed through the network towards the output layer. In this study, two forms of the so-called feed-forward ANNs are used, Multi-Layer Perceptrons (MLP) and Radial Basis Functions (RBF) [11]. Key transformation features of MLPs and RBFs is the multiplication of *weights* (weights express the strength of connections between neurons) and linear additions of *biases* i.e. during a calibration procedure known as training, inputs of a neuron are multiplied with the weight that accompanies their connection; the results are summed and an additional value (bias) is commonly added to this value. The resulting net input is transformed by some transfer function into an activation value of the neuron.

This activation value is then propagated to subsequent neurons. For a MLP, the relationship between the input and the output can be expressed as follows [11]:

$$y_k = \left[\sum_{j=1}^{M} w_{kj}^{(2)} f_{inner} \left(\sum_{i=1}^{d} w_{ji}^{(1)} x_i + w_{j0}^{(1)} \right) + w_{k0}^{(2)} \right] \tag{1}$$

where:

M represents the number of hidden units

d represents the number of inputs units

$w_{ji}^{(1)}$ and $w_{ji}^{(2)}$ represent weights in the first and second layer respectively, going from input *i* to hidden unit *j* and

$w_{j0}^{(1)}$ represents the bias for the hidden unit *j*.

For a RBF 2-layer network equation with *n* centres is expressed as follows [11]:

$$f_k(x) = \sum_{j=0}^{n} w_{kj} \phi_j(x) + b_k \tag{2}$$

where:

f_k represents the k-th output layer transfer function

ϕ_j represents the j-th input layer transfer function

and *w* and *b* represents the weights and biases

The input layer transfer function ϕ_j is a Gaussian transfer function as follows [11]:

$$\phi_j(x) = \exp\left(-\frac{\left\| x - \mu_j \right\|^2}{2\sigma_j^2} \right) \tag{3}$$

where:

x represents the input layer transfer function

μ represents the fixed centre position and

σ represents fixed variance.

A critical issue in ANNs is generalisation, and that is, how well an ANN performs with out-of-sample data. Too much training data and/or training cycles may lead to the ANN learning the noise and losing its generalisation capability (over-training). Too little training data and /or cycles may lead to the ANN failing to learn all the necessary information contained in the data set (under-training).

It must be noted that experiments performed in this study were conducted using both MLPs and RBFs. However, in this paper only results pertaining to MLPs are presented. This is because results for both were very similar; hence results pertaining to RBFs were omitted for the sake of succinctness and brevity.

4 AGENT DESIGN METHODOLOGY

4.1 The Multi-Agent Framework

In a stock trading environment, stock traders may, under certain circumstances, have a need to trade, manage, and track groups of stocks as one entity. This type of trading system is commonly referred to as *Basket Trading.* Basket traders enjoy the benefits of being able to personalise investment criteria (e.g. create baskets by sector and investment styles) and achieve greater diversification of investments. In Section 4.2 and 4.3, an agent is designed and optimised for use in a multi-agent basket-trading framework. From a Distributed Artificial Intelligence (DAI) point of view, an agent must simultaneously have at least the following main characteristics [12]:

- It perceives the world in which it is situated
- It has the capability of interacting with other agents
- It is proactive in the sense that it may take initiative and persistently pursues its own goals

Among others, the field of DAI deals with agent technology hence I appreciate that each agent feature mentioned in [12] may constitute an entire research study. Therefore in fairness to researchers in DAI, this study focuses only on the construction and “pruning” of the artificial stock prediction engine for an agent. This renders characteristics such as agent communication, mobility, etc, outside the scope of this work.

For the purposes of this study, an investors’ basket of stocks is imagined to be operated on by a multi-agent program, which allocates a single agent for each stock in the basket. Each agent contains an intelligent engine that is constructed based on two design qualities:

1. The ability to optimally analyse the statistical information contained in the particular stock’s time series data.
2. The ability to use the statistical information for a one-step prediction of the next day’s closing price

The former is achieved by simulating the evolution of a population of agents; who use technical analysis tools to generate decision signals (i.e. buy, sell, and hold signals). The latter is achieved by optimising an ANN that performs the actual prediction based on the technical analysis data determined from the fittest agent parameters yielded by the evolution process.

4.2 Optimising Technical Indicators

4.2.1 Optimising through profits as fitness

This part of the study uses a GA to simulate a multi-agent trading environment based on the work mostly done in [13], with a few exceptions. In our simulations: each agent is provided with the same starting capital and they are allowed to trade for 56 days using SABs closing prices, with the aim of maximising profits at the end of each 56 day trading cycle. Each agent trades the same stock since the goal is to eventually find an optimised agent for that particular stock. Fifty-six days of trading defines one generation of trading agents. Typically, at the end of each trading day, an agent performs technical analysis and analyses the stock using technical indicators. The mechanics behind stock trading, the complete genome used by the GA including the mathematical formulae for the technical indicators can all be studied in appendix A.

Technical indicators used in this study were: Moving Average (MA), Relative Strength Index (RSI), Bollinger Bands (BB), Stochastic Oscillator (SO), Price Rate-of-Change (ROC), and Moving Average Convergence Divergence (MACD). For each day, the value(s) returned by each of the technical indicators are interpreted in accordance with technical analysis theory [14] and a buy, sell, or hold signals are generated. The signals are then compiled for every agent and an overall buy signal is generated if *BUY SIGNALS* > *SELL SIGNALS* or vice versa and a hold signal if they are equal.

Each technical indicator mentioned above uses *n* amount of past closing price days to compute its value. For example a simple MA is computed using the following formula:

$$MA = \frac{\left(\sum_{j=1}^{n} price(j)\right)}{n} \tag{4}$$

where $n = \beta G_i$ for a short-term MA and $n = \alpha G_i$ for a long-term MA. The parameters β and α are weighted multipliers and G_i is the i-th parameter (gene) of interest to be optimised by the GA. Again the complete genomes for each agent including the decision logic used for the various indicators were derived from the studies performed in [13] and [15]. To save the effort of locating the papers, they are included in appendix A.

A population of 500 evolved over 100 generations is used for training. Fitness was determined by the total assets held by each agent at the end of every generation (i.e. the sum of its capital and value of all shares at the last day’s closing price). To form a new population, arithmetic; heuristic; and simple crossovers were used, while boundary; multi-non-uniform; non-uniform; and uniform mutations were used to alter single individuals. Tournament selection was used to select the best individuals in a population. Figure 2 below shows the results obtained from the simulations:

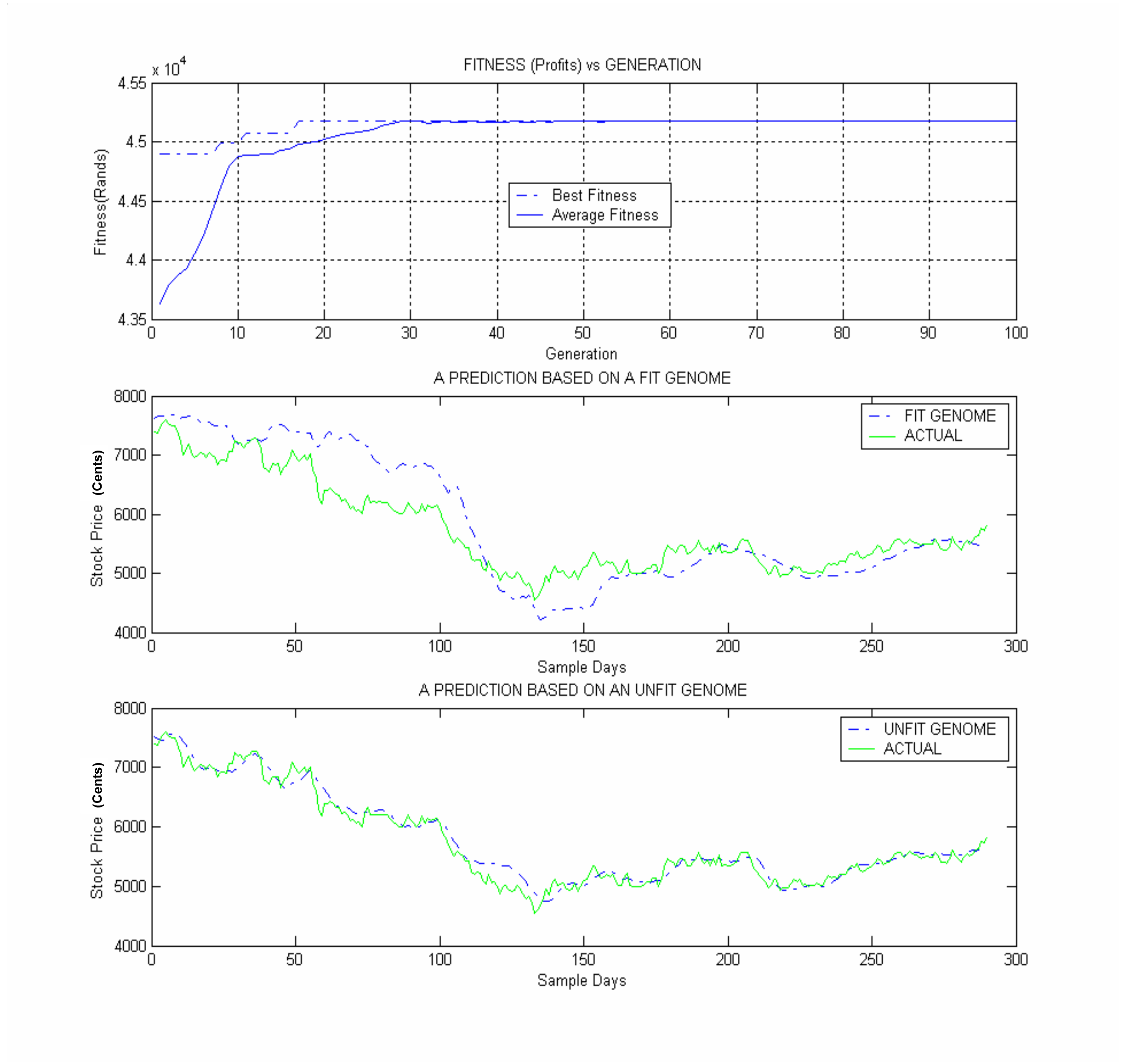


Figure 2. Evolution and prediction results obtained from comparing genomes that were evolved using profit as fitness.

As can be seen in Figure 2, the best population converges after 30 generations. It was anticipated that the genome yielded by the GA would represent optimal parameters needed to determine technical indicators for signal generation and, hence use the genome to generate a data matrix that can be used to train an ANN that will perform a one-step prediction. This process was then used to test the hypotheses that the most profit-making agent will on average make the best predictions. To test this; the most profit-making genome was compared to 50 other random genomes. From each of these genomes a data matrix with dimensions 6 × 900 (row × column) was generated and used to train a 2-layer MLP ANN to make one-step predictions. The MLP had 15 hidden nodes, 6 inputs, 1 output and was trained over 500 epochs with 600 days of data, validated and tested with 150 days of data each. The results showed that after 12 weeks of trading, the most profit-making agent did not necessarily make the best predictions. In Figure 2 the performance in one of the runs shows a randomly chosen (unfit) genome from generation 10 performing superior to an ‘optimal’ (fit) genome from generation 100. This meant that using profit as a fitness criterion did not necessarily optimise the technical indicators for predictive use and hence could not be used to optimise the technical indicators.

4.2.2 Optimising through predictions as fitness

A second alternative that uses validation predictions as fitness was explored. In this alternative I used a similar experiment to the one performed in Section 4.2.1 except for the following: In the evaluation function of the GA, instead of using the genome to compute technical indicators that generate trading signals, I used the genome to generate technical indicators that were immediately used to train an MLP network. Within the evaluation function, the validation data was forward-propagated and the GA was used to minimise the Mean Square Error (MSE) of the predictions.

The results of the experiment are shown in Figure 3. Again, 50 random genomes were compared with the best genome this time with the fit genome performing better predictions than the unfit genome. The experiment was repeated several times with randomly chosen genomes from various generations to check for consistency.
Based on the results obtained through this approach, the fit genome evolved using predictions as fitness was then selected as the genome that an agent will use each time technical indicators are generated for use as inputs to an ANN. Both experiments were performed using RBF ANNs with similar results being obtained.

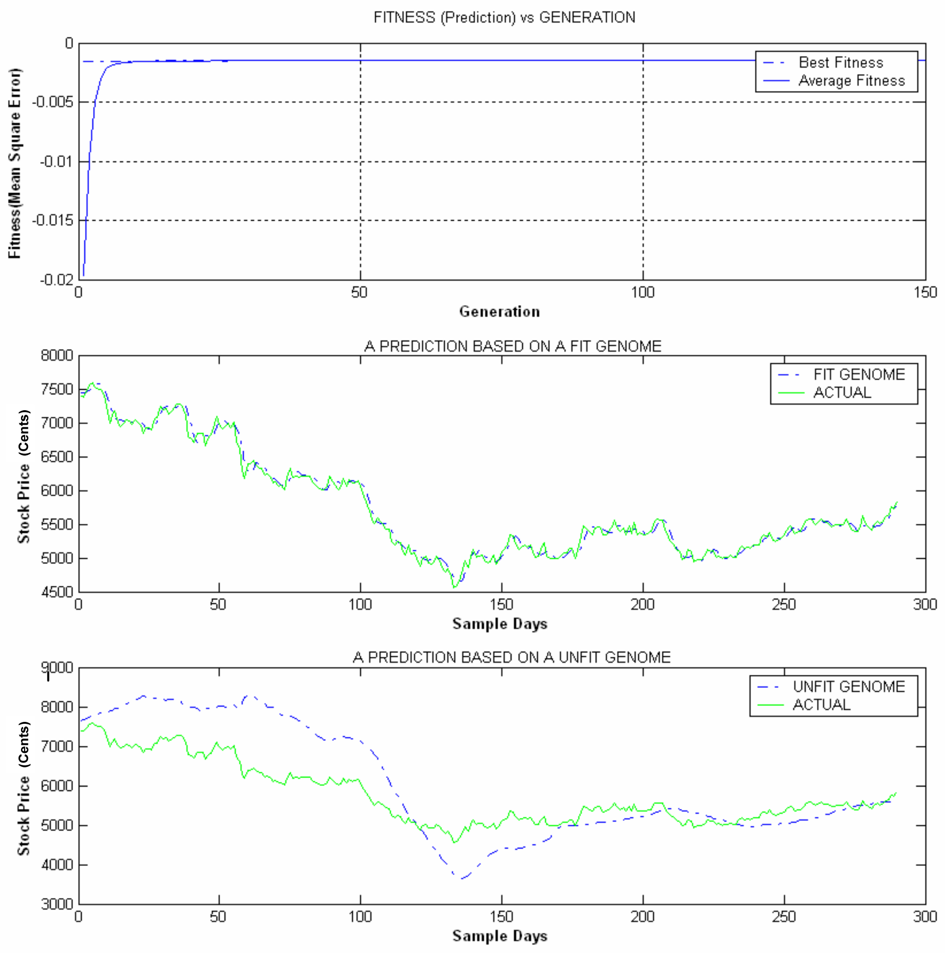


Figure 3. Evolution and prediction results obtained from comparing genomes that were evolved using MLP predictions as fitness.

4.3 Optimising the ANN Architecture

In Section 4.2, technical indicators that are used as inputs to the ANN were optimised. In this section, the interest is only in optimising the ANN engine that performs the prediction tasks. This is done by arriving at the best architectural attributes of the ANN. In general, network topology, neuron characteristics, and training or learning rules specifies ANN models. In [16], a distillation of the literature's best practice criteria for evaluating ANNs was outlined. In the study, two criterions were defined and are used in our study to design an architecture that will maximise generalisation capabilities: Effectiveness of validation and Effectiveness of implementation. Effectiveness of validation is concerned with the issue of whether a study has appropriately evaluated the predictive capabilities of the proposed network. On the other hand, effectiveness of implementation is concerned with whether a study implemented the ANN in such a way that it stood a chance of performing in accordance with benchmark results or even better.

To evaluate effectiveness of validation, three guidelines described in [16] are applied:

1. Comparisons with well accepted models
2. Use of ex ante (out-of-sample) validations
3. Use of a reasonable (75 or more) sample of forecasts

Guideline 1 will be satisfied by comparing our results with popular regression models as well as results obtained from naïve extrapolation (random walk). Using an out-of-sample data set of 150 validations satisfies guideline 2 and 3.

Effectiveness of implementation is evaluated using guidelines for evaluating network performance suggested by Refenes in [17]:

4. Convergence
5. Generalisation
6. Stability

Convergence will be ensured by reducing the error (MSE) during training to an acceptable value. Generalisation capabilities will be evaluated by ensuring that validation errors are similar to training errors. Stability is the consistency of results; using multiple samples for training and validation could ensure this.

A GA was used to design the architecture such that the above criteria are adequately met. The following were the parameters of interest being optimised by the GA: number of hidden nodes, number of training cycles, learning rate, and the training data size. These parameters were allowed to range from 2 – 40, 300 – 1000, 0.001 – 1, and 150 – 900 respectively. A backpropagation algorithm with the scaled conjugate gradient optimisation method was used for training with the GA minimising the MSE of the validation samples. An MLP and a RBF were optimised except that with the RBF the learning rate was omitted as a parameter of interest.

A population of a 100 ANNs was evolved over 60 generations for both networks. Convergence was reached at least around the 9th generation. This fact helps for future purposes when processing time becomes an issue. Critically, the amount of training data chosen by the GA was such that there was enough data to test and validate the network, hence obeying guideline 3.

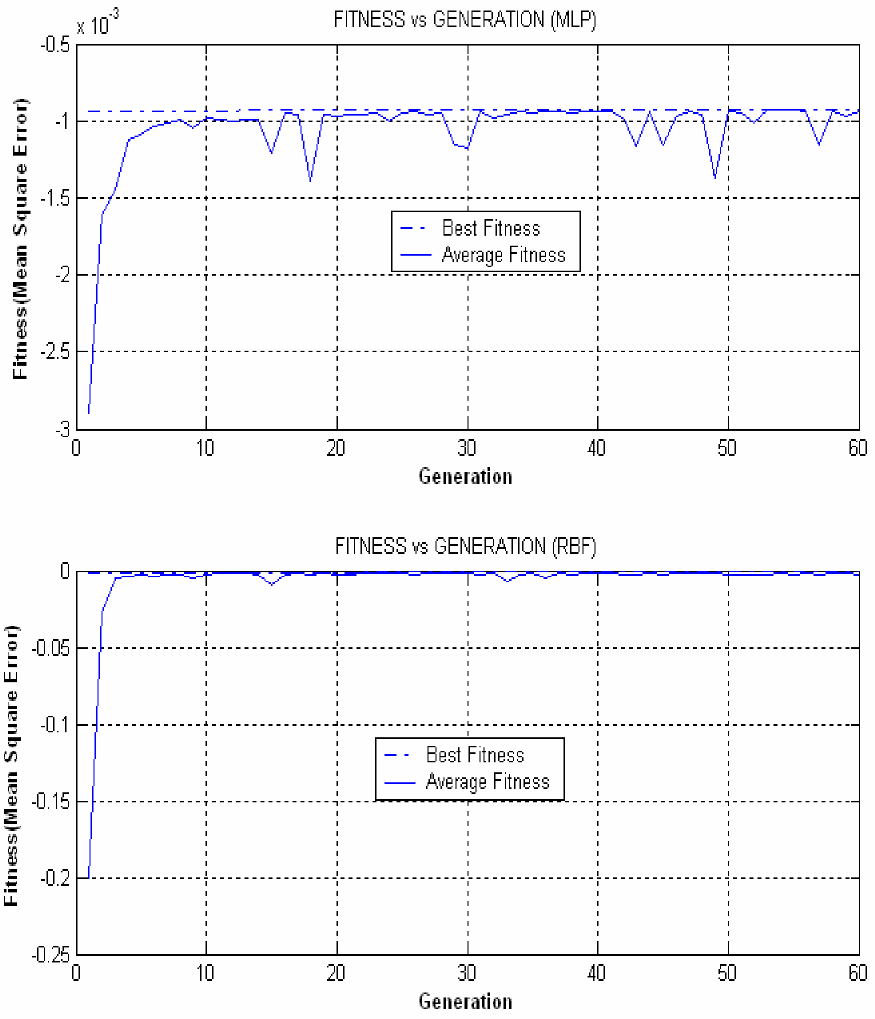


Figure 4. The evolution of the MLP and RBF architectures

5 MAIN RESULTS

5.1 Improving Predictions

The combined validation and testing prediction results shown in Figure 3 seem to have converged well for the fit genome because the trend is generally captured. Even the MSE indicated that the results were within the acceptable range. However; a zoomed visualisation at the plot reveals the existence of a prediction lag effect. By zooming into the plot we found that the lag was around 4 days. This means that the network predicted that the next day's closing price would be around what the price was 4 days ago. This lag we refer to as timing errors. The issue of timing errors in ANN forecasts has been rarely discussed in literature despite the many studies that have reported it [2], [18], and [19].

As an effort to deal with the timing errors, I decided to use additional input signals with the hope that the lag may be reduced or eliminated altogether. This technique assumes that additional and relevant information can contribute to the reduction of the observed prediction echo. As a result, past values comprising $t - 1$to $t - 4$ were used as additional inputs to ANN. The ANNs now had 10 inputs and the validation and testing results are shown in Figure 5.

After introducing additional input signals, the timing error was reduced to 1 day. The improvement can also be noted by the minimum and maximum prediction deviations for both ANNs: it can be seen that the deviations in prediction price for the 10 input network are lower than those of the 6 input network. This improvement is also significant in terms of the MSEs but still impractical for practical purposes. Section 5.2 discusses the cause of this echo effect and tries to offer two possible methods of interpreting the lagged predictions. A theoretical solution to eliminating the lagged predictions is also presented but because of its anticipated drawbacks; the training scheme is adjusted to yield better results.

It must also be noted that up to this point, the non-stationary nature of the time series is not considered. This aspect is of great importance because it can affect the accuracy of the predictions. By catering for non-stationarity, it is anticipated that the prediction results can be improved – the extent with which they can be improved is beyond the scope of this work. However, this problem can be countered by introducing an evolutionary *adaptation* of the technical indicators and the ANN architecture through a GA.

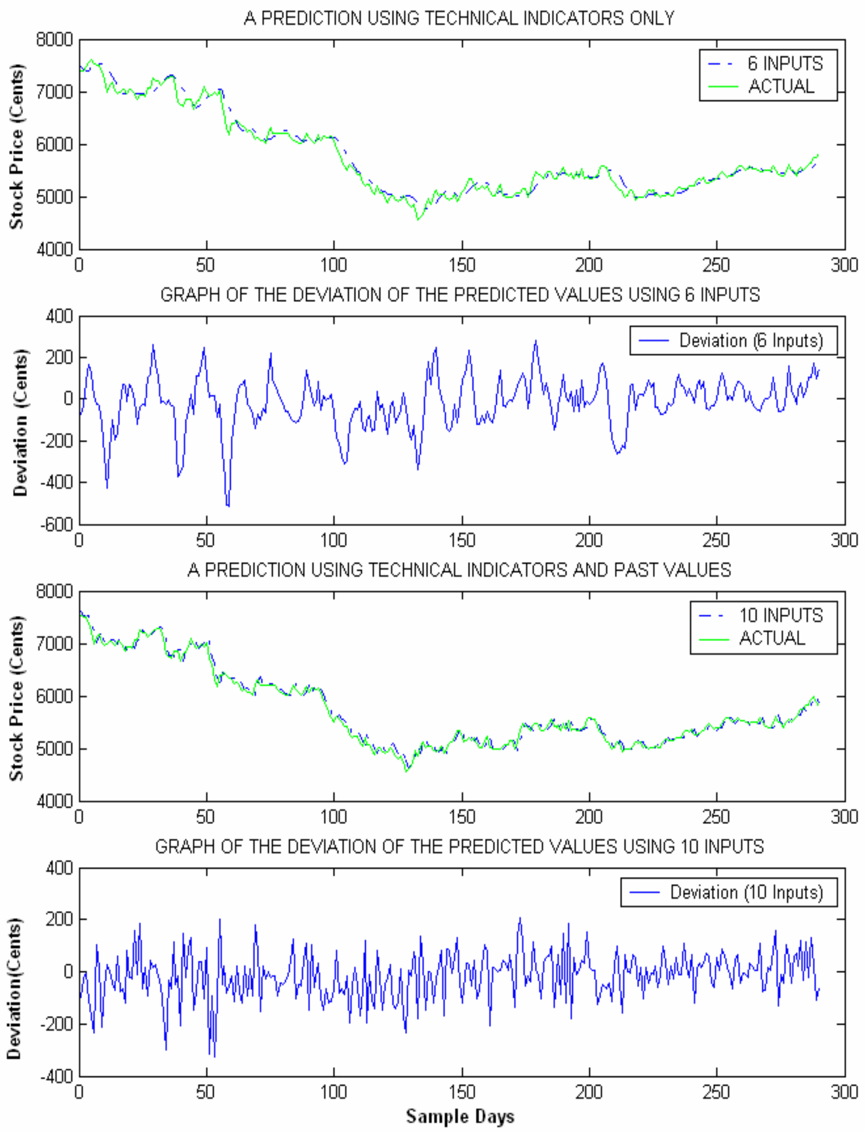


Figure 5. The effect of introducing additional input signals in order to improve predictions

5.2 Interpreting Predictions

The problem of timing errors raises the question of whether this failure to predict prices on time is the fault of the network, or is because the nature of the time series does not allow timely predictions to be made. In [19], a detailed study of non-linear auto-regressive time series models and linear moving average type time series showed that it was always possible for the best possible prediction schemes for these types of time series models to exhibit a time delay. The argument was that the high autocorrelation, of previously observed time series values, ensures that the auto-regressive model component, which is implicitly contained in ANNs, become dominant. The network gives the most weight to the latest value in order to predict the next value. In our case, this means that the network says that the best prediction for the next day's closing price is around the value of the currently observed closing price. Autocorrelation was performed on our time series and the results as shown in Figure 6 indeed indicate that the high autocorrelation could account for an implicitly dominant auto-regressive model component.

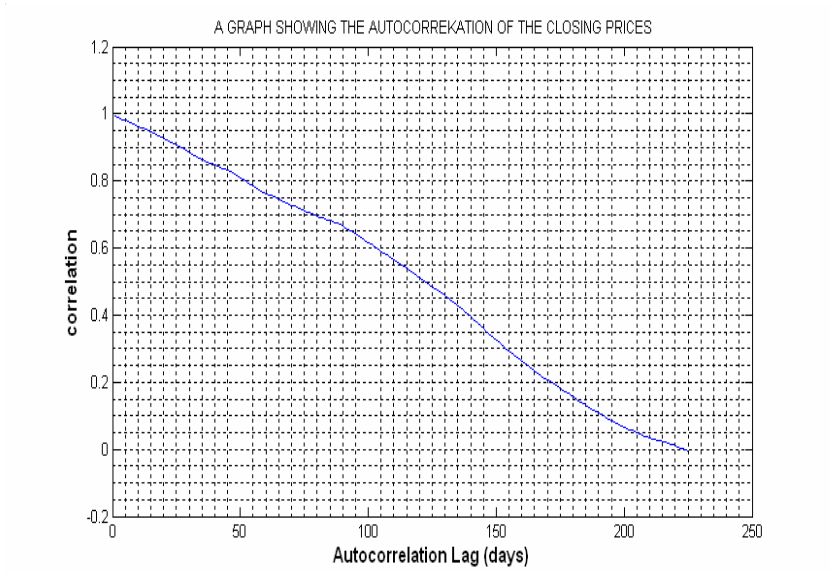


Figure 6. The autocorrelation of SAB's time series stock price data

As shown in Figure 6, the autocorrelation function decays after 225 days suggesting that after 225 days, little or no information about the present closing price is contained in the time series. Evidently the most recent price at lag zero has a correlation coefficient close to 1 hence supporting the argument that the most recent price will be given the most weight by an inherent auto-regressive component.

From this it is important to note that a performance measure such as MSE may be misleading so far as the correct timing of predictions is concerned. In Conway's study using ANNs to predict sunspots [19]; it was asserted that the delay effect is attributable to a reliance on MSEs, since this is the natural criterion that backpropagation networks use to select the best predictions. For possible solutions to this problem, four approaches are explored:

The first approach uses a Hit Rate for performance measurement as defined in [21]. There Qian and Wah showed that noise did not contribute to prediction accuracy and therefore needed to be removed. Therefore their problem involved the prediction of a smoothed (low-pass version) signal that lagged behind actual price changes. Their hit rate metric was defined as follows: Let $D(t+h) = sign(S(t+h) - S(t))$ be the actual direction of change for $S(t)$, and $\hat{D}(t+h) = sign(\hat{S}(t+h) - \hat{S}(t))$ be the predicted direction of change. A prediction for a step h is a hit if $\hat{D}(t+h) * D(t+h) > 0$ and hence the hit rate is defined by the following equation:

$$H(h) = \left| \frac{\{D(t+h)\hat{D}(t+h) > 0\}}{\{D(t+h)\hat{D}(t+h) \neq 0\}} \right| \text{ for } t = 1,\ldots,n \qquad (5)$$

Figure 7 illustrates the definition. Simply, two thresholds are specified based on the lag period; these thresholds are shown as *$S(t_0)$* and *$\hat{S}(t_0)$* in Figure 7. Whenever both signals are above or below their respective thresholds, for example $t\varepsilon[t_0,t_1];[t_2,t_3]$, it is a hit. In contrast, when the signals are on different sides of their respective thresholds, $t\varepsilon[t_1,t_2];[t_3,t_4]$, it is not a hit.

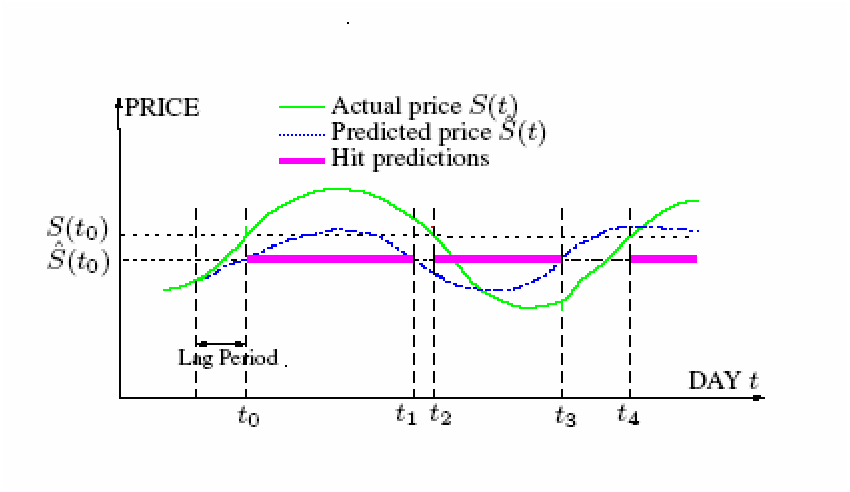


Figure 7. An illustration of how hits are interpreted in predictions [21]

In this study, the same hit rate metric was implemented except that the time series were not converted to low pass versions. However, this method has its drawbacks: the hit rates are dependent on the time at which the signal is being observed. The results showed that hit rates for combined validation and testing samples were as high as 85% and as low as 34%. By observing predictions in Figure 5, it is easy to see that thresholds defined anywhere between samples 0 to 100 will interpret all samples 100 to 290 as hits. These results would be clearly misleading. For more meaningful results, a region of 20 samples that exhibited little volatility in fluctuations was chosen. The hit rate achieved in this region was 72%. This method and hence the results are not seen as holistically objective because there is no set rule for defining where in time, the most objective thresholds are. Later in this Section, an improved metric for measuring performance in terms of percentage Direction Success is introduced.

A second approach involves trading strategies used by Yao and Tan in a study whose predictions also experienced lags [2]. Their strategies were as follows:

Strategy 1

if $(\hat{x}_{t+1} - \hat{x}_t) > 0$ then *buy* else *sell*

Strategy 2

if $(\hat{x}_{t+1} - x_t) > 0$ then *buy* else *sell* where x_t is the actual price at time *t*, and $\hat{x}_t$ is the predicted price at time *t*. Yao and Tan assert than the reason for the different strategies lies in the 'noise' ζ_t such that $x_t = \hat{x}_t + \zeta_t$. They mentioned that the more accurate strategy depended on $|\zeta_{t+1} - \zeta_t|$. If ζ_{t+1} and ζ_t have different signs, strategy 1 is better than strategy 2, otherwise the opposite holds. In their paper, Yao and Tan did not quantify what this noise is or even how it is measured. Hence I borrowed strategy 1 in its pure form without the noise term. The sense in strategy 1 is that a trader will perceive stock fluctuations with reference to his past predictions. In other words, over a period of time, a trader's time series fluctuations would be replaced by that of predictions. This has the advantage that even if there is a one-day lag in the predictions, the ANN will be able to capture periods of sharp rises or falls. However, during periods of significant fluctuations, but less spiky, trading decisions could still lag. This means that this approach must also be seen as half-effective.

The third approach draws on the work done by Conway in [20] and is only discussed theoretically in this paper. As mentioned earlier, Conway attributed the timing error to an inherent performance measure used by the backpropagation algorithm. In this respect, our lagged predictions indicate that the MSE performance measure failed to penalise the timing errors of the predictions. It follows then that it is important to use multiple performance measures (commonly termed multi-objective calibration) during training. In his Thesis, Conway adapted a neural network-training scheme so that the error measure considers the RMS error and a penalty for delayed predictions. In his work, a GA was used to arrive at the best network. His simulations were similar to our simulations in Section 4.2, but his fitness was determined by the network with low RMS error and no delay. His results were positive in that the delay had disappeared but with a significant increase in RMS error. A succinct description of the trade-off between delay and RMS error is also presented in [22]. For our study, this is a conundrum because both the timing accuracy and RMS error are critical. Conway's technique is yet to be tested on our work but, due to the expected high RMS errors, it is anticipated that the results would be undesirable in practice.

5.3 Further Improving Predictions and Performance Measurement

From the three approaches discussed in Section 5.2, unfortunately none proves adequate for satisfactorily dealing with prediction lags. However, Conway's multi-objective approach can be seen as the better of the three approaches. In this Section a fourth approach that hybridises the first and third approaches is suggested [23]. Here a multi-objective calibration scheme that improves predictions and provides an objective performance measurement technique is explored.

The first approach is advantageous in that it is trying to measure performance by quantifying how well the predictions predict the correct direction of the stock's movement for the next day. An example of this importance is illustrated in Figure 8.

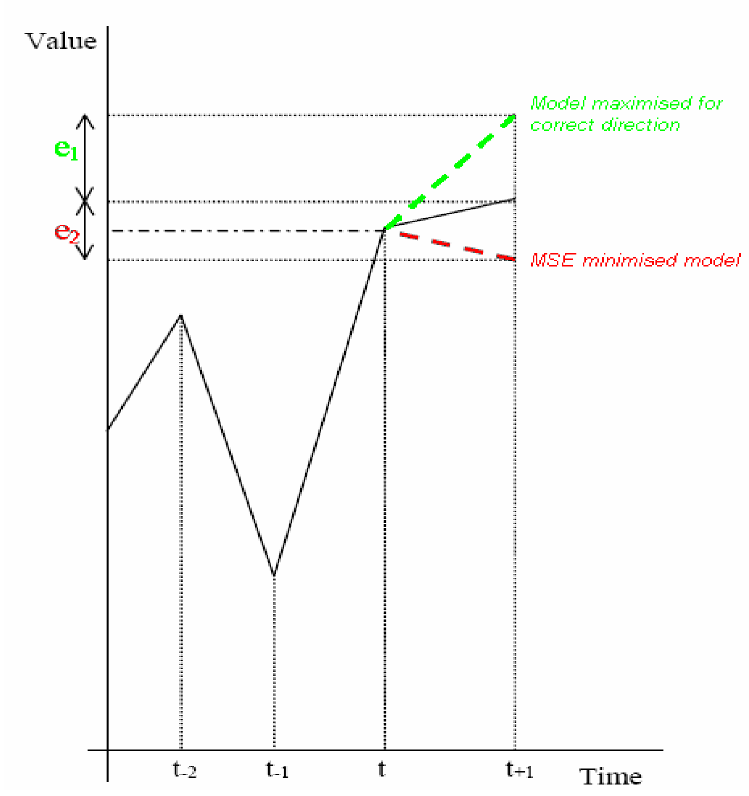


Figure 8. A comparison between a model maximised for correct prediction of direction and a model that is MSE minimised [23].

In Figure 8 it can be seen that the model with error e_1 will have a higher MSE than the model with error e_2. However, intuitively the model with error e_1 is preferable since it correctly predicts the direction of movement for the next day, although the MSE may be sacrificed.

To this effect, a multi-objective technique that optimises directional success and MSE is suggested. But as mentioned earlier, the method of measuring directional success using the technique depicted in Figure 7 was seen as unreliable; therefore a different and a more reliable set of formulae for calculating percentage direction success error (DSE) [23] are shown in equations 6, 7, and 8:

$$DSE = 100 - \left(\frac{\sum_{t=1}^{P} H_t}{P} .100 \right) \qquad (6)$$

$$\Delta \hat{y}_t = \hat{y}_{t+1} - \hat{y}_t \qquad (7)$$

$$\Delta y_t = y_{t+1} - y_t \qquad (8)$$

In equation 6, $H_t = 1$ if predicted ($\Delta \hat{y}_t$) and actual (Δy_t) changes are the same sign, otherwise $H_t = 0$.

The multi-objective approach will attempt to minimise the MSE and the DSE. A GA is used in a similar fashion to the simulations performed in Section 4.3 to optimise an architecture that minimises the MSE and DSE. The GA was set-up to minimise the objectives in the following fashion:

$$Fitness = (\beta_1 * (-DSE)) + (\beta_2 * (-MSE)) \qquad (9)$$

where $\beta_1 > \beta_2$ so that the DSE has greater weight and is given greater preference in determining fitness. A 100 MLP networks were evolved over 600 generations and the results are shown in Figure 9.

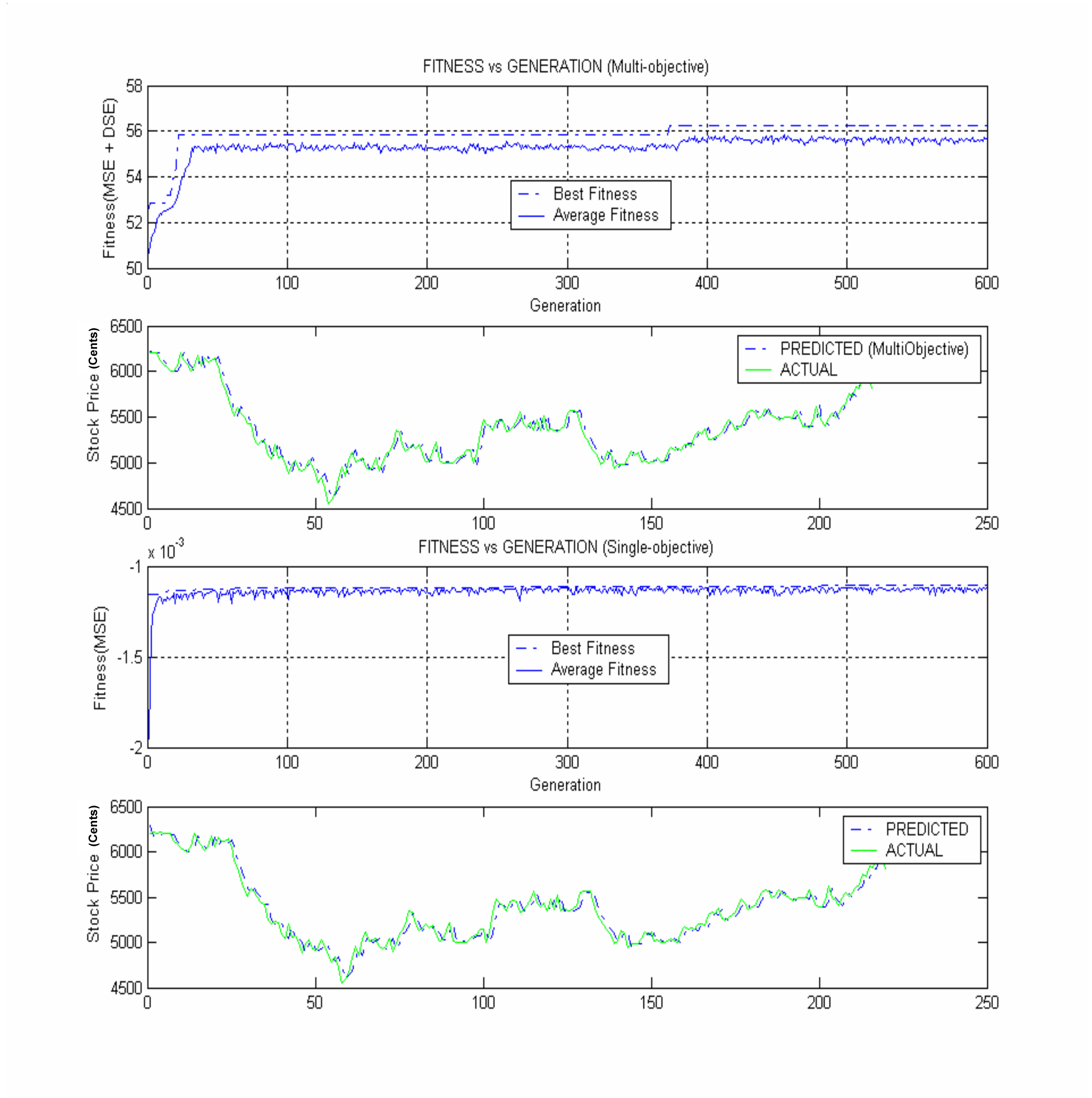


Figure 9. The comparison of evolution results between a multi-objective approach and a single objective approach.

As expected, there was a trade-off between DSE and MSE. In the simulations, the multi-objective simulations achieved higher percentage direction successes (between 54% and 77%) than the single-objective simulations (between 48% and 55%). On the other hand, the single-objective simulations achieved higher MSEs than the multi-objective simulations. By observing Figure 8, it can be noted that the multi-objective fitness can be still be improved by increasing the number of generations and intuitively, the population size.

This multi-objective approach is lucrative in that the timing error can also be incorporated in the fitness function. Further work in this study will hence explore the possibility of optimising the architecture around a fitness function that incorporates three objectives where the third objective is a quantified timing error (TE). In this scenario, the following fitness could be used to derive an architecture that has no timing errors and minimised MSEs and DSEs: $Fitness = (\beta_1 * (-DSE)) + (\beta_2 * (-MSE)) + (\beta_3 * (-TE))$

6 CONLUSIONS

In this paper, a multi-agent framework in which an intelligent engine for an agent is constructed is proposed. A GA is used to test the hypotheses that the most profitable trader in a specified time-span will always perform better predictions than others based on a genome that optimised technical indicators. From the simulations it was shown that this hypothesis was not true and hence the genome could not be seen as reliable. However, caution needs to be exercised because traders were simulated over a 56-day trading span. It is our view that simulations over much longer trading days may yield different results. The best genome was however arrived at by using validation MSEs as fitness. This genome was used to generate an input data matrix consisting of technical analysis data that would be used to train an ANN.

A GA was again used to optimise the ANNs architecture with hidden nodes, training cycles, learning rate, and training data size as parameters to be optimised. The predictions showed a significant lag of 4 days but after including additional input signals in the form of past values, the lag was reduced to one day. This prediction lag is caused by the high autocorrelation of the time series, hence facilitating for an implicit dominant autoregressive component.

Four approaches from other literature for dealing with the lag were discussed: the hit rate approach is dependant on where in time the thresholds are defined and it was advised that a stable region should be used to compute the hit rate using this technique. A second approach suggested using only the predictions to determine whether a buy signal or a sell signal is generated. It was concluded that this approach will be most beneficial during periods of sharp fluctuations and therefore is also limited. A third approach suggested a multi-objective performance measurements scheme during training. This method exhibits a trade-off between timing accuracy and RMS error and it is anticipated that in practice it would be rendered impractical. A fourth approach explored a hybrid technique that uses an improved measure of quantifying DSE along with MSE in a multi-objective calibration environment. This hybrid approach is recommended as a framework under which an agent’s ANN architecture can be optimised.

Further work in this study will explore the possibility of incorporating timing errors in the objective function such that the delay in predictions can be eliminated while DSEs and MSEs are minimised. Another issue involves the non-stationary nature of the time series. It is recommended here that an adaptation mechanism be incorporated in the system to adapt the ANN inputs as well as the ANN architecture.

7 REFERENCES

[1] Lawrence R. Using Neural Networks to Forecast Stock Prices. University of Manitoba, 1997.

[2] Yao J., Tan C.L. A Case Study on Using Neural Networks to Perform Technical Forecasting of Forex. www.elsevier.com/

[3] Hurst H.E. Long Term Storage Capacity of Reservoirs. Trans. Am. Soc. Civil Eng. 116 (1951) 770-808.

[4] Peters E.E Chaos and Order in the Capital Markets: A New View of Cycles, Prices, and Market Volatility. Wiley, New York, 1991.

[5] Box G.P.E., Jenkins G.M. Time Series Analysis: Forecasting and Control. Holden Day, San Francisco, CA, 1976.

[6] Bnsal A., Kauffman R.J., Weitz R.R. Comparing the Modeling Performance of Regression and Neural Networks as Data Quality Varies: A business Approach, *Journal of Management Information Systems,* Vol 10. pp 11-32, 1993

[7] Marquez L., Hill T., Worthley R., Remus W., Neural Network Models as an Alternate to Regression, *Proc. Of IEEE 24th Annual Hawaii Int'l Conference on System Sciences,* pp.129-135, Vol VI, 1991.

[8] Man-Chung C., Chi-Cheong W., Chi-Chung L. Financial Time Series Forecasting by Neural Network Using Conjugate Gradient Learning Algorithm and Multiple Linear Regression Weight Initialisation. The Hong Kong Polytechnic University.

[9] Davis L. The Handbook of Genetic Algorithms. Van Nostrand Reingold, New York, 1991.

[10] Holland J. Adaptation in Natural and Artificial Systems. The University of Michigan Press, Ann Arbor, 1975.

[11] Bishop C M. Neural Networks for Pattern Recognition. Oxford University Press. Twelfth Edition, 1995.

[12] Oliveira E., Fischer L., Stepankova O. Multi-Agent System: Which Research for which Applications. Robotics and Autonomous System, 1999, pp. 91-106.

[13] Schoreels C., Garibaldi J.M. A Preliminary Investigation into Multi-Agent Trading Simulations Using a Genetic Algorithm. University of Nottingham.

[14] Achelis S.B. Technical Analysis from A to Z. McGraw-Hill Trade, 2000.

[15] Schoreels C., Garibaldi J.M. The Effects of Varying Parameters on Performance for Adaptive Agents in Technical Equity Market Trading. *IEEE 3rd International Conference on Computational Cybernetics.* Mauritius, 2005.

[16] Adya M., Collopy F. How Effective are Neural Networks at Forecasting and Prediction? A Review and Evaluation. *Journal of Forecasting*. 1998, pp. 481-495.

[17] Refenes A. N., Azema-Barac M., Zapranis A. D. Stock Ranking: Neural Networks vs Multiple Linear Regression. *IEEE International Conference on Neural Networks*, 1993, pp. 1419-1426

[18] Jain A., Srinivasulu S. Development of Effective and Efficient rainfall-runoff models using intergration of Deterministic, Real-Coded Genetic Algorithms and Artificial Neural Network Techniques. Water Resour. Res., 40(4), W04302, 2004.

[19] Conway A. J., McPherson K.P., Brown J.C. Delayed Time Series Predictions with Neural Networks. *NeuroComputing 18,* 1998, pp. 81-89.

[20] Conway A. J. The Prediction and Analysis of Solar Terrestrial Time Series. Ph.D Thesis, University of Glasgow, 1995.

[21] Wah B.W., Qian M. Constrained Formulations for Stock Price Predictions Using Recurrent FIR Neural Networks. American Association for Artificial Intelligence, 2002.

[22] Conway A.J. Echoed Time Series Predictions, Neural Networks and Genetic Algorithms. Vistas in Astronomy 38 (3), 1994, pp. 351-356.

[23] Training Neural Networks beyond Euclidean Distance, Multi-Objective Neural Networks using Evolutionary Training. http://www.smartquant.com/references/NeuralNetworks/neural26.pdf . Last accessed 18 November 2005.

[24] Stock Basics: Introduction. http://www.investopedia.com/university/stocks/ . Last accessed 06 December 2005.

[25] Cybenko G. Approximation by Superpositions of a Sigmoidal Function. Mathematics of Control, Signals, and Systems. 1989, pp. 303-314.

[26] Michalewicz Z. Genetic Algorithms + Data Structures = Evolution Programs. AI Series. Springer – Verlag, New York. 1994.

[27] Joines J., Houck C. On the Use of Non-Stationary Penalty Functions to Solve Constrained Optimisation Problems with Genetic Algorithms. *IEEE International Symposium on Evolutionary Computation.* Orlando. pp. 579-584, 1994.

[28] Wooldrige M., Jennings N. R. Intelligent Agents: Theory and Practice. *Knowledge Engineering Review.* 1995.

APPENDICES

APPENDIX A

STOCK TRADING

1 INTRODUCTION TO STOCK TRADING

1.1 Overview

This appendix deals with the area of financial investments through stock trading. The first part of this document provides an introduction to stock trading for the layman. The second part of the document describes the tools that were used for the stock trading simulations that were performed in Section 4 of the main document. The area of financial investment is vast and there exists many different forms of financial investments. For the sake of relevancy and succinctness, this appendix only concerns itself with stock trading. This introduction follows the introductory template used in [24].

The history of stock trading dates back to before early-mid nineteenth century. Since then and up to recent years the average person's interest in this type of investment has grown immensely. Coupled with the advent of technological advances such as computers and the Internet, nearly every money-oriented individual can own and trade stocks. However, despite the growing interest in stocks, many people still don't understand how stock markets function let alone the risks involved with this type of investment 'game'.

In this introduction we will start by defining what a stock is; highlight the different types of stocks that exist; how stocks are traded; what influences their price fluctuations; how to buy stocks; and lastly the different trader personalities found in the stock markets.

1.2 Stocks, Shares, Equities; What are they?

Simply put, a stock is ownership of a corporation as evidenced by shares, which are a claim on the corporation's earnings and assets. In other words, the more stocks an entity acquires, the more ownership that entity has in the company of interest. In the context of stock trading, the terms *share* and *equity* mean the same thing.

Stock ownership is traditionally represented by a piece of paper that serves as proof of ownership for that particular stock – this paper is called a stock certificate. However recent advances in computer technology enabled the electronic maintenance of these certificates such that seamless trading is facilitated.

It must be noted that being a shareholder does not mean that one has a say in the day-to-day running of the corporation. Instead, with each share, a shareholder has a vote in the election of professional individuals (called a board of directors) who maintain the running of the company. The board of directors ensures the smooth profitable maintenance of the company and if this does not happen, the shareholders may replace the board accordingly. It is not common though that every shareholder owns enough shares to influence key transformations of a company. It is usually large institutional shareholders and billionaires who have a say into which individuals will run the company.

It is not very concerning that individual shareholders do not have a say in the running of the company because of the limited liability involved with owning stocks. For example with companies such as partnerships, everybody involved in the partnership is liable if the company goes bankrupt. Above all, the greatest advantages of owning stocks are that shareholders make money through the stock’s appreciation and dividends paid out by the company. Although the latter is not compulsory for companies, the latter is usually enough to generate decent yields.

It should be understood that like all business, trading stocks involve risk. In the case where a company goes bankrupt or the stock price drops significantly, great monetary losses are incurred. On the bright side, taking on greater risks demands greater returns.

1.3 The Different Types of Stocks

Two main types of stocks that exist are the *common stock* and the *preferred stock.* The main difference with the stocks is their representation of the type of ownership interest in a corporation. Common stockholders assume the greater risk, but generally exercise the greater control and may gain the greater award in the form of dividends and capital appreciation in the long run. However if the company goes bankrupt the common stockholders will not receive their money until the creditors and preferred stockholders have received their respective share of the leftover assets. On the other hand a preferred stock is a type of stock that pays a fixed dividend regardless of corporate earnings, and which has priority over common stock in the payment of dividends especially in the event of liquidation of a company. It carries no voting rights, and should earnings rise significantly the preferred holder is stuck with the same fixed dividend while common holders collect more. The fixed income stream of preferred stock makes it similar in many ways to bonds.

Common stocks and preferred stocks are the main types of stocks but it is also possible in some instances to have different classes of stocks. The common reason for this is if the company wants to assign different voting powers to certain groups. For example a certain group of shareholders may have 20 votes per share while another group has 10 votes per share and so on.

1.4 Buying and Selling Stocks

Stocks are traded at the stock exchange like the Johannesburg Stock exchange (JSE) where buyers and sellers meet and decide on bid and ask prices. Stock exchanges are either a physical location where transactions are carried out on trading floor or virtual, where transactions are carried out electronically supported by a network of computers.

Two types of markets are distinguished, the ‘primary’ and the ‘secondary’ markets. The primary market is where a company initially creates stocks, whereas in the secondary market traders exchange previously issued stocks without the involvement of the issuing companies.

Popular stock exchanges are the New York Stock Exchange (NYSE) where trading is predominantly performed on the trading floor, the NASDAQ where most of the trading is performed via internetworked computers, the London and Hong Kong Stock Exchanges and in London and Hong Kong respectively.

1.5 Stock Fluctuations

The fundamental reason the stock price fluctuates is because of supply and demand. This point is easy enough to understand, the difficulty is in comprehending the underlying dynamics that govern people's preferences to certain stocks. What is certain though is that the movement of a stock's price is a reflection of what investors feel the company is worth at the time. At any time, the value of a company is its market capitalisation which is the stock price multiplied by the number of shares outstanding. Furthermore, the stock price is a reflection of what investors expect the stock to be worth in the future.

The leading factor that affects a company's stock price is its earnings. This is the reason why public companies are required to report their earnings several times a year. Earnings are important in that analysts base the future value of the company on the earnings projection – if a company's earnings are better than expected, the jumps otherwise the opposite holds.

1.6 How to Buy Stocks

There are two main ways of buying stocks. The first and most common way is buying through a brokerage entity and the second is through Dividend Reinvestment Plans (DRIPs) or Direct Investment Plans (DIPS). The major difference between the two is that brokerage forms offer advice and guidance while DRIPs and DIPS do not. Brokerage firms offer various types of options and DRIPs and DIPS also come in myriads of ‘flavours’, the details of which are beyond the scope of this work.

1.6 Stock Trading Personalities

Stock trading as a ‘discipline’ has its own extensive jargon. One such jargon is the vocabulary used for the different types of trader personalities found on the stock markets. The following are used for various descriptions: Bulls, Bears, Chickens, and Pigs.

A bull market is when the economy is thriving and stocks are subsequently rising. If a trader is optimistic that stocks will go up, they are referred to as “Bulls”. A bear market is when the economy is bad, GDP is dropping, and recession is anticipated. If a trader is pessimistic and believes that stocks are going to drop, they are called a “Bear”. Chickens are overly cautious traders who never want to rake risks. Chickens never see any significant returns because of their fear to try things out. Pigs are high-risk traders who rely on hot tips to make quick big scores without doing their due-diligence. They are impatient and often greedy which usually leads them to grave losses. Bulls and Bears make most of their money from Pigs.

1.7 Conclusions

Stocks imply ownership of a portion of the company. It is important to note that it is conceivable that one can lose all their investments in stocks with the upside being that lots of money can be made if the right investments are made. Common stocks and Preferred stocks are the two main types of stocks but certain companies may feel the need to create different classes. Traders exchange stocks at the stock market and this can either be done on a trading floor or via internetworked computers. While supply and demand determine the price of stock, other factors such as earnings and investor confidence play a role. To buy stocks, either a brokerage firm or DRIPs/DIPS can be used. Lastly, Bulls and Bears make money at the expense of the Pigs.

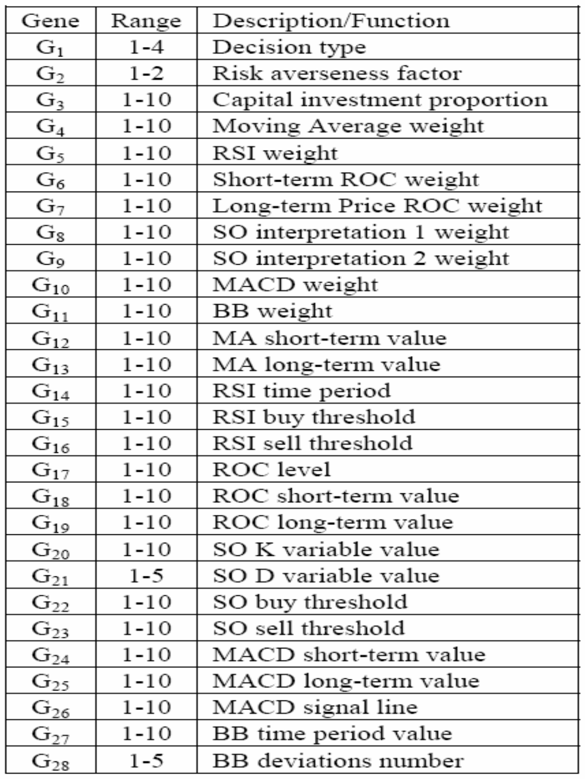
2 STOCK TRADING SIMULATIONS

2.1 Technical Indicators

This section is meant to provide greater detail on the simulations performed in Section 4 in the main document, and is borrowed from the work done in [13]. As mentioned in the main document, there are two main types of analyzing stocks, Fundamental Analysis and Technical Analysis. In this section, we outline the technical formulae used for the stock prediction task. Section 2.2 of appendix A furthers this discussion by providing the decision logic that was used for the indicators used in the simulations.

It was discussed that a Genetic Algorithm (GA) was used to optimise parameters for technical indicators such that the optimised technical indicators can be used as input data for an artificial neural network (ANN). The genome for each individual in the population is shown in Table 1 below.

Table 1. Gene Description



Gene	Range	Description/Function
G_1	1-4	Decision type
G_2	1-2	Risk averseness factor
G_3	1-10	Capital investment proportion
G_4	1-10	Moving Average weight
G_5	1-10	RSI weight
G_6	1-10	Short-term ROC weight
G_7	1-10	Long-term Price ROC weight
G_8	1-10	SO interpretation 1 weight
G_9	1-10	SO interpretation 2 weight
G_{10}	1-10	MACD weight
G_{11}	1-10	BB weight
G_{12}	1-10	MA short-term value
G_{13}	1-10	MA long-term value
G_{14}	1-10	RSI time period
G_{15}	1-10	RSI buy threshold
G_{16}	1-10	RSI sell threshold
G_{17}	1-10	ROC level
G_{18}	1-10	ROC short-term value
G_{19}	1-10	ROC long-term value
G_{20}	1-10	SO K variable value
G_{21}	1-5	SO D variable value
G_{22}	1-10	SO buy threshold
G_{23}	1-10	SO sell threshold
G_{24}	1-10	MACD short-term value
G_{25}	1-10	MACD long-term value
G_{26}	1-10	MACD signal line
G_{27}	1-10	BB time period value
G_{28}	1-5	BB deviations number

The following technical indicators were used:

- Moving Average (MA)
- Relative Strength Index (RSI)
- Bollinger Bands (BB)
- Stochastic Oscillator (SO)
- Price Rate-of-Change (ROC)
- Moving Average Convergence Divergence (MACD)

The short-term and long-term MA values are calculated as follows:

$$MA = \frac{\sum_{i=1}^{N} price(i)}{N} \tag{1}$$

where price(i) is the current closing price of a stock, $N = 4G_{12}$ and $N = (5G_{12}) + 50$ for the short-term and long-term respectively.

The Relative Strength Index (RSI) is a technical analysis indicator that measures relative gains, over relative losses, over time. The RSI is calculated as:

$$A = \frac{\sum_{i=1}^{N} UpCloses}{Number\_of\_Upcloses}$$

$$B = \frac{\sum_{i=1}^{N} DownCloses}{Number\_of\_Downcloses}$$

$$RSI = 100 - (100/((A/B0+1) \tag{2}$$

where $N = 2.5G_{14}$

The Price Rate-of-Change (ROC) indicator assumes cyclical price movements and considers the relative change of prices over time to indicate trends. It is calculated as follows:

$$ROC = price(i) - price(i+N) \tag{3}$$

where $N = 2G_{18}$ and $N = 4G_{19}$ for the short-term and long-term respectively.

The Stochastic Oscillator (SO) is a momentum indicator that measures the price of a security relative to the high/low range over a set period of time. It is calculated as follows:

$A = price(i) - Lowest\_Close$

$B = Highest\_Close - Lowest\_Close$

$$K = (A / B) * 100 \tag{4}$$

$$D = (\sum_{i=1}^{N} K(i)) / N \tag{5}$$

where $N = G_{21}$

The Moving Average Convergence Divergence (MACD) is an indicator that follows the difference between a series of moving averages. The indicator has two lines, the MACD line and a signal. A buy signal is generated when the MACD line rises above the signal line. A sell is generated when the MACD line fall below the signal. Because the MACD is generated from moving averages it is has a unique ability capture wide swinging moves in markets. Divergence, trendlines and support lines can also be applied to the MACD to generate additional signals. The MACD is calculated:

$$MACD = \exp MA(2G_{24}) - \exp MA(4.5G_{25}) \tag{6}$$

where

$$\exp MA = ((price(i) - prevMA) * (2 / (N + 1))) + prevMA \tag{7}$$

where N represents the time period it was measured over $G_{24} or G_{25}$ days and the prevMA is the previous exponential moving average apart from the first instance, where a simple moving average is used.

Lastly, Bollinger Bands (BB) plot trading bands above and below a simple moving average. The standard deviation of closing prices for a period equal to the moving average employed is used to determine the bandwidth. This causes the bands to tighten in quiet markets and loosen in volatile markets. The bands can be used to determine overbought and oversold levels, locate reversal areas, project targets for market moves, and determine appropriate stop levels. The upper and lower bands are calculated as:

$$stdDev = \sum_{i=1}^{N} (price(i) - MA(N))$$

$$upperBand = MA(3.5G_{27}) + (stdDev * G_{28}) \qquad (8)$$

$$lowerBand = MA(3.5G_{27}) - (stdDev * G_{28}) \qquad (9)$$

where $N = 3.5G_{27}$

2.2 Technical Analysis Decision Logic

The following were the decision logic used in the simulations. The decision logic was implemented exactly as they were implemented in [13].

For the MA:

```
BOOLEAN A = MA(4G12) < current price
BOOLEAN B = MA((5G13)+50) < MA(4G12)
BOOLEAN C = (G2 == 1)
IF ( A AND B AND C ) OR (NOT C AND ( A OR B )
Action: buy
ELSE
Action: sell
ENDIF
```

For the RSI:

```
IF RSI(2.5G14) >= (4G16)+50
Action: sell
ELSEIF RSI(2.5G14) <= 5G15
Action: buy
ELSE
Action: hold
ENDIF
```

For the ROC:

```
IF ROC(2G18) < -G17
Action: buy
ELSEIF ROC(2G18) > G17
Action: sell
ENDIF
```

For the SO:

```
IF K(1.5G20) > D(G21)
Action: buy
ELSE
Action: sell
ENDIF
```

For the MACD:

```
IF MACD(2G24,4.5G25) >
MA(1.5G26,MACD(2G24, 4.5G25))
Action: buy
ELSE
Action: sell
ENDIF
```

For the BB:

```
IF lowerBand(3.5G27) >= current price
Action: buy
ELSEIF upperBand(3.5G27) <= current price
Action: sell
ENDIF
```

APPENDIX B

ARTIFICIAL NEURAL NETWORKS

1 INTRODUCTION

Artificial Neural Networks (ANNs) have received increased interest in the past few years primarily because of their power and ease of use. ANNs are particularly powerful because they are nonlinear and deploy sophisticated modeling techniques that are capable of modeling extremely complex functions.

ANNs grew out of research in Artificial Intelligence (AI), specifically, attempts to mimic and loosely model the biological processes of a neuron (Figure 10). The brain contains in excess of 10,000,000,000 of these structures; each connected to some 10,000 other neurons. Neurons have a neuronal cell (Soma) with input and output channels (Dendrites and Axons) that connect them. Each neuron receives electrochemical input signals at the dendrites. If the sum of these electrochemical inputs is powerful enough to activate a neuron, it transmits an electrochemical signal along the axon (referred to as ‘firing’). This mechanism is propagated throughout the entire neural network with some neurons firing and others not.

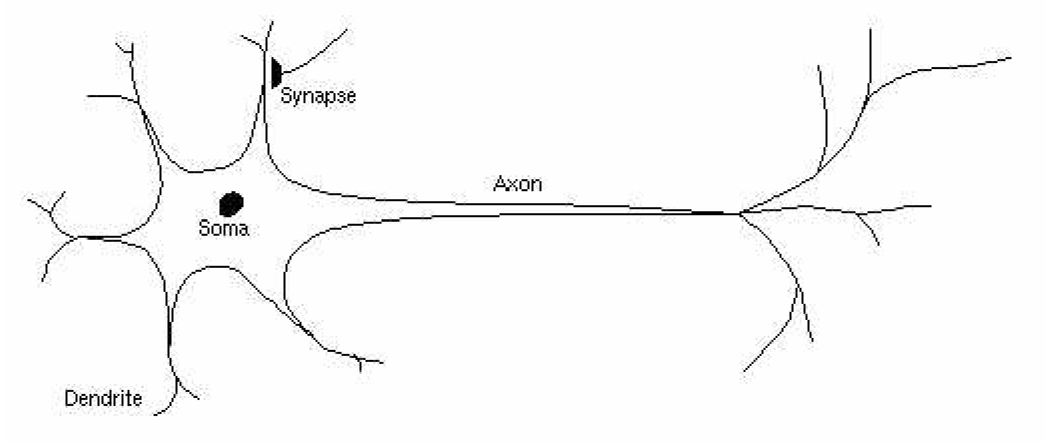


Figure 10. A simple biological neuron

To capture the essence of a biological neuron, an artificial neuron is defined as follows:

- It receives inputs via a connection that has strength (called *weights*). These weights correspond to the synaptic excitory or inhibitory effect in a biological neuron. Each neuron contains a threshold value. The weighted sum of the inputs is formed, and the threshold subtracted to compose the *activation* of the neuron.
- The activation signal is passed through an activation function that produces the output of the neuron.

An analogy of a simple artificial neuron is depicted in Figure 11. These artificial neurons are connected together in a layered fashion to create an ANN as shown in Figure 12. The blue circles denote neurons and the lines connecting them denote the weights.

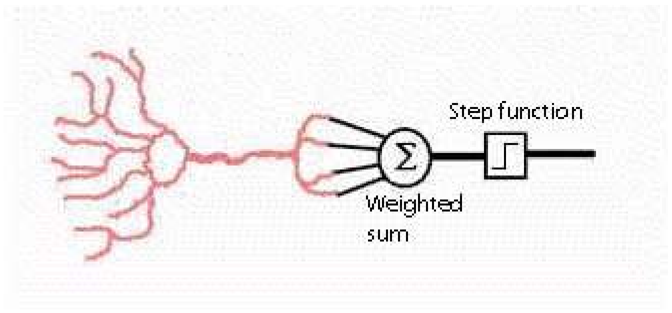


Figure 11. A depiction of how an artificial neuron relates to a biological neuron

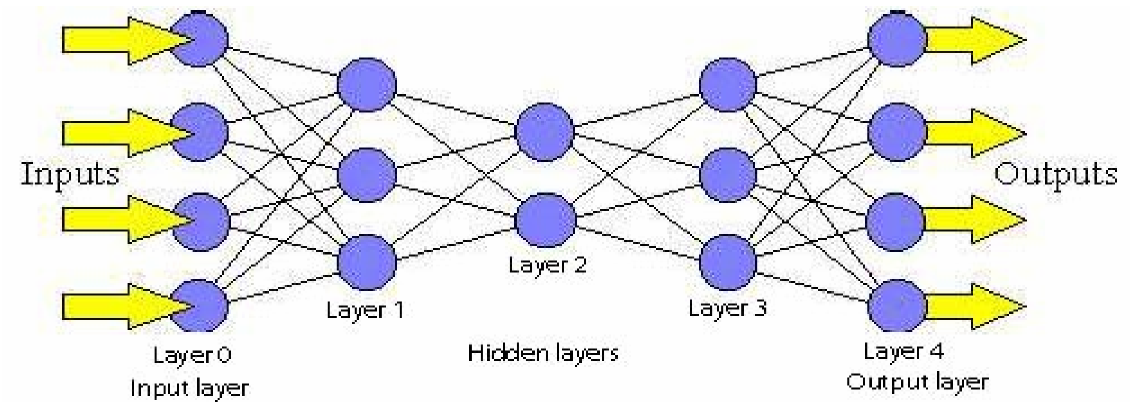


Figure 12. An artificial neural network

2 APPLYING NEURAL NETWORKS

Because ANNs work by manipulating inputs variables into output variables, they can be used in problem where there is some known information that could be used to infer unknown information. Some examples include prediction and control problems – fundamentally ANNs are primarily used to solve complex pattern recognition problems.

An important requirement when using ANNs is that the relationship between the proposed inputs and outputs is known. This relationship may be noisy and stochastic but it must exist. In general the exact nature of this relationship is unknown; otherwise it would be modeled directly. A key feature is that an ANN will learn the input/output relationship through a process called *training.* Two forms of training exist: supervised learning and unsupervised learning.

In supervised learning the network user assembles a set of training data that contains examples of inputs together with the corresponding outputs, and the network learns to infer the relationship between the two. The data used for learning is usually taken from historical records. Unsupervised learning consists of a set of training algorithms that adjust the weights in an ANN by reference to a training data set including input variables only. Unsupervised learning algorithms attempt to locate clusters in the input data.

In this appendix we concern ourselves with supervised learning since it was the training method used in this study

3 GATHERING DATA

Once a problem is formulated and it needs ANNs to solve, the next step is to gather the necessary data for training purposes. This process requires a decision to be made about which variables to use and how many cases will suffice. Choosing variables is generally guided by intuition coupled with some expertise in the problem domain. ANNs are designed to process numeric data so the common obstacles when dealing with data are unusual ranges of values (e.g. normalizing the data between -1 and 1), missing data, and non-numeric data. These obstacles must be overcome by scaling the data to an appropriate range for the network, missing data can be substituted by statistically plausible values, and non-numeric values can be intelligently converted into numbers (e.g. Male = 1 and Female = 2).

The number of training example can be considerably difficult to determine. There are some heuristics available but, fundamentally this number is related to the complexity of the underlying function we are trying to model. Furthermore, in [16] [17] it is suggested that enough data should be collected such there is enough training data, validation data, and testing data.

Another problem with data is that there may be training examples that are unreliable. Commonly these present themselves as outliers and hence need to be identified and removed.

4 NEURAL NETWORK ARCHITECTURES

The two most popular ANN architectures in use today is the Multi-Layer Perceptron (MLP) and the Radial Basis Function (RBF), with the former being the more popular of the two.

4.1 The Multi-Layer Perceptron

4.1.1 Architecture

The basic MLP network is constructed by ordering the units in layers, letting each unit in a layer take as input only the outputs of units in the previous layer or external inputs. If the network has two such layers, it is referred to as a two-layer network, if it has three layers it is referred to as a three-layer network. Due to the structure, this type of network is often regarded to as a *feedfoward* network. Figure 12 illustrates an MLP network structure. Important issues in MLP design include specification of the number of hidden layers and the number of units in these layers. However, in [25] it was shown that all continuous functions can be approximated to any desired accuracy, in terms of the uniform norm, with a network of one hidden layer of sigmoidal (or hyperbolic tangent) hidden units and a layer of linear output units. Once the number of layers, and number of units in each layer, has been selected, the network's weights and thresholds must be set so as to minimize the prediction error made by the network. The weights are adjusted using an optimisation algorithm that minimises an error surface obtained from comparing the Euclidean distance between the actual values and the predicted values. The best-known example of such an algorithm is the *backpropagation* algorithm. In our study we used a modernised second-order algorithm called the *Scaled Conjugate Gradient* (SCG) [11] descent algorithm.

For a MLP, the relationship between the input and the output can be expressed as follows [11]:

$$y_k = \left[\sum_{j=1}^{M} w_{kj}^{(2)} f_{inner} \left(\sum_{i=1}^{d} w_{ji}^{(1)} x_i + w_{j0}^{(1)} \right) + w_{k0}^{(2)} \right] \tag{1}$$

where:

M represents the number of hidden units

d represents the number of inputs units

$w_{ji}^{(1)}$ and $w_{ji}^{(2)}$ represent weights in the first and second layer respectively, going from input i to hidden unit j and

$w_{j0}^{(1)}$ represents the bias for the hidden unit j.

The following are common activation functions for MLPs:

$$linear : f(x) = x \tag{2}$$

$$\tan gent : f(x) = \tanh(x) \tag{3}$$

$$sigmoid : f(x) = \frac{1}{(1+\exp(x))} \tag{4}$$

$$step : f(x) = \operatorname{sgn}(x) \tag{5}$$

4.1.2 Scaled Conjugate Gradient

The SCG algorithm is an advanced method of training MLPs and can be used to substitute backpropagation because it usually outperforms backpropagation significantly. The SCG algorithm is a recommended technique for networks with a large number of weights and/or multiple output units whereas its closely linked variants, Quasi-Newton and Levenberg-Marquardt may be better for small networks with low-residual regression problems.

SCG is a batch update algorithm: whereas backpropagation adjusts the weights after each case, SCG works out the average gradient of the error surface across all cases before updating the weights once at the end of the training cycle. Furthermore, there is usually no need to select learning and momentum rates for SCG, so it can be much easier to use than back propagation.

SCG works by performing a series of line searches across the error surface. It first works out the direction of steepest descent in a similar fashion to backpropagation. However, instead of taking a step proportional to a learning rate, SCG projects a straight line in that direction of the minimum and then locates a minimum along this line. This process is not quick because it only searches for the minimum in one dimension. Subsequently, further line searches are conducted at every training cycle. The directions of the line searches, called the conjugate directions, are chosen to try to ensure that the directions that have already been minimised remain minimised.

It must be noted that the conjugate directions are calculated on the assumption that the error surface is quadratic, which is not always so. However, it is a fair working assumption, and if the algorithm discovers that the current line search direction isn't actually downhill, it simply calculates the line of steepest descent and restarts the search in that direction. Once a point close to a minimum is found, the quadratic assumption holds true and the minimum can be located very quickly.

3.2 The Radial Basis Function

A second type of ANN that finds popular use is the RBF. An RBF is a type of ANN that employs a hidden layer of radial units and an output layer of linear units with each hidden unit modeling a Gaussian response surface. At the input of each neuron, the distance between the neuron center and the input vector is calculated. Applying the basis function to this distance then forms the output of the neuron. The RBF network output is formed by a weighted sum of the neuron outputs and the unity bias. Since the basis functions are nonlinear, it is not necessary to have more than one hidden layer to model any shape of function: sufficient radial units are enough to model any function.

For a RBF 2-layer network equation with n centres is expressed as follows [11]:

$$f_k(x) = \sum_{j=0}^{n} w_{kj}\phi_j(x) + b_k \tag{6}$$

where:

f_k represents the k-th output layer transfer function

ϕ_j represents the j-th input layer transfer function

and *w* and *b* represents the weights and biases

The input layer transfer function ϕ_j is a Gaussian transfer function as follows [11]:

$$\phi_j(x) = \exp\left(-\frac{\left\|x-\mu_j\right\|^2}{2\sigma_j^2}\right) \tag{7}$$

where:

x represents the input layer transfer function

μ represents the fixed centre position and

σ represents fixed variance.

Because RBFs require only one hidden layer, they have an advantage over MLPs because this simplicity removes some design decisions. Secondly, the linear transformation in the output layer can be fully optimized using traditional linear modeling techniques. Linear modeling techniques are quick and do not suffer local minima which is not the case with

MLP training techniques. To this effect RBF networks can be trained in the order of magnitudes quicker than MLPs.

However RBFs requires significantly more units to adequately model most functions. Consequently, a RBF solution will tend to be slower to execute and more space consuming than an MLP. The choice between a RBF and a MLP architecture would be determined by the constraints of the problem: If faster training is a priority, then a RBF architecture is suitable; other wise if faster execution is a priority, then a MLP architecture is suitable.

APPENDIX C

1 GENETIC ALGORITHMS

1.1 Introduction

Simply put, Genetic Algorithms (GAs) belong to a family of evolutionary computational models inspired by the process of evolution and natural genetics. They practice survival of the fittest by evolving a set of encoded potential solutions (i.e. string structures) until the fittest solution is realized. Information is exchanged randomly but they efficiently exploit historical information to speculate new search points that exhibit improved ‘genes’. This feat is achieved by applying recombination operators to the string structures such that critical genetic information is preserved [10]. GAs are extensively used in optimisation problems whose objective functions are difficult and do not possess desirable properties such as continuity, differentiability, satisfaction of the Lipschitz Condition, etc. [9]

An implementation of GAs begins with a random population of solutions. These solutions are evaluated and allocated reproductive opportunities in such a way that the solutions that represent a better solution to the target problem are given more chances to be represented in the next generation than the poorer solutions. However, inferior solutions can, by chance, survive and also reproduce to the next generation. Through this GAs explore all regions of the state space and they exponentially explore promising areas through its recombination operators: mutation, crossover, and selection.

A GA is governed by six fundamental issues: solution representation, selection function, genetic operators and reproduction function, population initialisation, termination criteria, and the evaluation function. The rest of this appendix describes each of these issues.

1.2 Solution Representation

A GA requires an effective solution representation for each individual in the population of interest. Each solution is a string containing a sequence of genes from a particular alphabet. The alphabet may consist of binary digits, floating point numbers, integers, symbols, matrices, etc. In [10], Holland's representation was limited to binary digits but subsequent research by Michalewicz [26] showed that real-valued representations move the problem closer to the problem representation, which offers higher precision and consistent results. A sample genome representation can be viewed in Table 1 in appendix A.

1.3 Selection Function

The selection function in a GA is responsible for selecting the fittest individuals for the next generation. Selection is performed based on a probabilistic technique that ensures the fittest solution is represented in the next generation. There are several schemes for the selection process: roulette wheel, scaling techniques, tournament, elitist models, and ranking models. In this appendix we describe the roulette wheel, the ranking models and tournament selection because of their general popularity.

The roulette was developed by Holland [10] and it implements a biased structure where each current string in the population has a roulette wheel slot sized in proportion to its fitness. The probability P_i for each individual is defined by:

$$P_i = \frac{F_i}{\sum_{j=1}^{PopSize} F_j} \qquad (1)$$

where F_i is the fitness of individual i.

Ranking models assign P_i based on the rank of solution i when all solutions are sorted. Normalised geometric ranking [27] defines a P_i for each individual by:

$$P_i = q'(1-q)^{r-1} \qquad (2)$$

where:

q = the probability of selecting the best individual

r = the rank of the individual, where 1 is the best

P = is the population size

$$q' = \frac{q}{1-(1-q)^P} \tag{3}$$

Tournament selection does not assign probabilities; it works by selecting j random individuals, with replacement, from the population set, and inserts the best of j into the new population. The process is continued until N individuals are selected.

1.3 Selection Function

Genetic operators are the basic search mechanism of the GA. Here some members of the population undergo transformations by means of unary and higher order transformations. The former creates new individuals by a small alteration in a single individual whereas the latter creates new individuals by breeding (crossover) parts from several individuals. After some specified generations (iterations), the program converges and it is hoped that the fittest individual from that population represents a near-optimum solution. An example simple crossover is shown below:

Given two parent chromosomes, green and red,

10001001110010010
01010001001000011

Choose a random bit along the length, say at position 9, and swap all the bits after that point so that the above becomes:

10001001101000011
01010001010010010

There exist different types of crossover techniques and the reader can study them extensively in [26]

1.5 Initialisation, Termination, and Evaluation Functions

A GA needs to be seeded with an initial population. A common method is to randomly generate an entire population, however, to speed up the process the beginning population can be seeded with potentially good solutions.

The GA evolves the solutions from generation to generation until a termination criterion is met. Commonly, the GA is terminated after a specified number of generations. Another criterion involves population convergence criteria. Because GAs generally forces the entire population towards a single solution, when the sum of the deviations among the individuals becomes smaller than specified threshold, the algorithm can be terminated. The termination criteria mentioned here are not exhaustive hence several other hybrid strategies can be implemented.

The evaluation function is the driving force behind the GA. This function is used by the GA to calculate the fitness of each solution. An evaluation function is unique to the optimisation problem and therefore a different evaluation function must be developed for each different problem.

APPENDIX D

1 A BRIEF INTRODUCTION TO MULTI-AGENT SYSTEMS – AN EXTENDED DEFINITION

In this appendix I provide an overview of Multi-Agent Systems (MAS). Agent technology is a very active field of Distributed Artificial Intelligence (DAI), a large area of study. With this in mind, this appendix will not deal with in-depth details because MASs are extremely problem-domain specific. Instead, this appendix should be viewed as more of an extended definition of Agents and MASs. Agent technology has a strong philosophical and mathematical background; for those readers who are interested in these in-depth aspects, excellent work has been conducted by Wooldridge and Jennings (among others) and their paper in [28] is a good start.

Before any discussion on MASs, it is perhaps necessary to define what an Agent is. Several researchers have proposed formal definitions and we retain the following notional definitions described in [28]: A weak and a strong notion of agency.

A weak notion of agency is characterised by the following properties:

- *Autonomy*: agents operate without the direct intervention of humans or others, and have some kind of control over their actions and internal state.
- *Social ability*: agents interact with other agents (and possibly humans) via some kind of *agent-communication language.*
- *Reactivity*: agents perceive their environment, and respond in a timely fashion to changes that occur in it.
- *Pro-activeness*: agents do not simply act in response to their environment; they are able to exhibit goal-directed behaviour by *taking the initiative*.

For some researchers in Artificial Intelligence (AI), the term 'agent' is conceptualised and implemented using concepts that are usually applied to humans; they characterise an agent using *mentalistic* notions such as knowledge, belief, intention, and obligation. This notion is perceived as a stronger definition, and the following are various other attributes in the context of agency:

- *Mobility* is the ability of an agent to move around an electronic network.
- *Veracity* is the assumption that an agent will not knowingly communicate false information.
- *Benevolence* is the assumption that agents do not have conflicting goals, and that every agent will therefore always try to do what is asked of it.
- *Rationality* is (crudely) the assumption that an agent will act in order to achieve its goals, and will not act in such a way as to prevent its goals being achieved — at least insofar as its beliefs permit.

In general, an agent's computational structure will contain the following:

- Static knowledge on itself and on other agents (acquaintances).
- Expertise knowledge that represents treatments and actions that an agent is able to carry and which can be described in various forms (production rules, frames, logical expressions, etc.).
- Reasoning: the inferences which draw the problem resolution.
- Communication; the communication protocols between the agents.
- Cooperation strategies used by the agents to cooperate with others.
- Mobility: the ability to transport itself from one machine to another in an intelligent manner, retaining its current state.

The abovementioned notions try to define an agent; a MAS can then be seen as system composed of a population of autonomous agents which corporate with each other towards a common goal. For a MAS to function coherently, the agents must communicate among themselves and coordinate activities. Without this functionality, agents become a collection of individual entities that may (when in large numbers) become chaotic.

There are several reasons why agents need coordination and good communication:

- Preventing chaos: No agent can posses a global view of the entire MAS it belongs, as this is infeasible in any community of reasonable complexity. Consequently, agents have only local views, goals and knowledge that may interfere with rather than support other agents' actions. Coordination is vital to prevent chaos during conflicts.
- Agent's actions are frequently interdependent and hence an agent may need to wait for another agent to complete its task before executing its own. Such interdependent activities need to be coordinated.

Another important component of agent scheduling is the communication protocols among agents. In order to achieve this coordination, the agents might have to interact and exchange information; therefore they need to communicate by sending messages. KQML (Knowledge Query and Manipulation Language) is a good example of agent communication language.

In conclusion to this definition, MASs find an immense use in almost every area of distributed computing from air-traffic control to telecommunications. Agent-based modeling is another area of AI that uses agent technology to better understand complex structures such as the social behaviour of animals and the spreading of viruses.

MAIN CONCLUSION

This research work has dealt with ANNs and GAs in the effort to use their machine learning power to better approximate future events. As a minimum, the work can be seen as lower-bounds for unstructured-decision making.

Perhaps the most appealing aspect of ANNs in this research study was their ability to learn, rather than construct a symbolic model of the world. It emerged that ANNs rely heavily on the plasticity (i.e. distributed representation) of their own structure to adapt directly to experience. Because of this, I feel that learning is among the most important aspects of intelligence. Although on-line learning was not implemented in this study, it was recommended and hence the pressing question of whether effective learning can occur with no prior or initial knowledge. The answer to this question is drawn from the experience of constructing the ANN architectures themselves. Hindsight suggests that some sort of prior knowledge, usually expressed as an inductive bias, is necessary for learning in complex environments. This is because the ability of neural networks to converge on a meaningful generalization from a set of training data has proven sensitive to the number of artificial neurons, the network topology, and the specific learning algorithms used in training. Researchers have commented on the matter of selecting appropriate numbers of input values, the ratio between input parameters and hidden nodes, and the training trials necessary before convergence can be expected. Another issue of importance is the quality and quantity of data. Without extensive and built-in knowledge of the problem-domain, a learning algorithm can be totally misled attempting to find patterns in noisy, insufficient, or bad data. A GA implementation in this study incorporated all these factors as parameters of interest in a vast optimisation problem, with successful results.

On the other hand, genetic and emergent models of computation offered this research an approach to understanding both human and artificial intelligence. By demonstrating that globally intelligent behaviour can arise from the corporation of large numbers of restricted, independent, embodied, individual agents, this work has shown that genetic and emergent theories can address the issue of complex intelligence expressed in the interrelationships of relatively simple structures. In this work, this was done by simulating a multi-agent trading environment that helped in choosing the best attributes of a trader. In other words, using an agent-based approach, I have constructed an informative miniature model of how the various dynamics relate in trading markets. This was successfully done via a GA because the algorithm offers a powerful and flexible search of a problem space. It was seen that this is because a genetic search is driven both by diversity enforced by mutation and by operators, such as crossover, that preserve important aspects of parental information for succeeding generations.

The results of this research are promising, and further work is anticipated over and above the techniques that are being used in this work. Finally, all learning paradigms are tools for empirical study. As we further explore the invariants of our universe, we begin to probe into questions related to the nature of perception, understanding, learning, and problem solving.