CENTRE FOR EMEA BANKING, FINANCE & ECONOMICS

GP Algorithm versus Hybrid and Mixed Neural Networks

Andreas Karathanasopoulos

Working Paper Series

No 16/11

GP Algorithm versus Hybrid and Mixed Neural Networks

Andreas Karathanasopoulos
London Metropolitan University

A.Karathanasopoulos@londonmet.ac.uk

Abstract

In the current paper we present an integrated genetic programming environment, called java GP Modelling. The java GP Modelling environment is an implementation of the steady-state genetic programming algorithm. That algorithm evolves tree based structures that represent models of input – output relation of a system. The motivation of this paper is to compare the GP algorithm with neural network architectures when applied to the task of forecasting and trading the ASE 20 Greek Index using only autoregressive terms as inputs. This is done by benchmarking the forecasting performance of the GP algorithm and 6 different ARMA-Neural Network combination designs representing a Hybrid, Mixed Higher Order Neural Network (HONN), a Hybrid, Mixed Recurrent Network (RNN), a Hybrid, Mixed classic Multilayer Perceptron (MLP) with some traditional techniques, either statistical such as a an autoregressive moving average model (ARMA), or technical such as a moving average convergence/divergence model (MACD), plus a naïve trading strategy. More specifically, the trading performance of all models is investigated in a forecast and trading simulation on ASE 20 time series closing prices over the period 2001-2008 using the last one and a half years for out-of-sample testing. We use the ASE 20 daily series as many financial institutions are ready to trade at this level and it is therefore possible to leave orders with a bank for business to be transacted on that basis.

As it turns out, the GP model does remarkably well and outperforms all other models in a simple trading simulation exercise. This is also the case when more sophisticated trading strategies using confirmation filters and leverage are applied, as the GP model still produces better results and outperforms all other neural network and traditional statistical models in terms of annualised return.

1. INTRODUCTION

The use of artificial intelligence for the purpose of forecasting market movements has been widely reviewed in academia. This study is a comparative analysis of the results yielded from utilizing a Genetic Programming Algorithm and various traditional Neural Network computing techniques when forecasting the Greek stock market. Additionally, we endeavour to develop more accurate and sophisticated techniques in order to increase the performance of our trading simulation. Due to the convergence and unification of global financial markets in recent years, this forecasting task has become increasingly challenging. Furthermore, traditional econometric methods on which forecasters have previously been reliant no longer satisfy the demands of market participants as they struggle to capture integrating features associated with today's markets. As discussed by Lisboa *et al* (2000), neural networks are an emergent technology with an increasing number of real-world applications offering a unique aspect to the world of financial forecasting. Nevertheless, some practitioners have tainted the virtues of neural networks with scepticism criticising their capacity to forecast and highlighting their limitations. Hence, this paper investigates a new, contemporary and more proficient method of forecasting that is capable of identifying and dealing with discontinuities, nonlinearities and high frequency multi-polynomial components which are all prevalent in financial series of today's markets. This model is most commonly known as the Genetic Programming (GP) algorithm.

GP Algorithms are domain-independent problem-solving techniques that are run in various environments. These environments are structured in a manner which approximates problems in order to produce forecasts at a high level of accuracy. GP can be categorized in the forecasting bracket known in the finance world as 'Evolutionary Algorithms'. The basis for this type of problem – solving technique derives from the Darwinian principle of reproduction and *survival of the fittest.* Additionally, GP is also similar to the biological genetic operations such as crossover and mutation. More importantly, Koza (1990, 1992) stress that GP addresses and quantifies complex issues as an automated process via programming, which enables computers to process and solve problems.

The Darwinian aspect of GP applies the theory of evolution to a population of computer programs of varying sizes and shapes. For instance, GP starts with an initial population of thousands or even millions of randomly generated computer programs. These programs comprise of programmatic elements built to apply the fundamental principles of biological evolution in order to create a new (and often improved) population of programs. As mentioned previously, the creation of this new population is generated in a domain-independent system applying the Darwinian theory of natural selection under the principal known as *survival of the fittest*. An analogue of the naturally-occurring genetic operation of sexual recombination (crossover), and occasional mutation, the crossover operation is designed to create syntactically valid offspring programs (given closure amongst the set of programmatic ingredients). GP combines the expressive high-level symbolic representations of computer programs with the near-optimal efficiency of learning of Holland’s (1975) genetic algorithm in order to produce highly accurate outputs. Koza (1998) mentioned that a computer program that solves or at the very least approximates a given problem often emerges from this process. Dissimilar to other models such as neural networks, GP does not require any prior knowledge of a model’s structure for the purpose of system modelling. Alternatively, GP evolves a system model with parameter values that best fit specific data without manipulating the data to fit ‘predefined’ model structures as many other preceding forecasting methods tend to do. In other words, GP creates an initial population of models and evolves using genetic operators in order to calculate the mathematical expression which best fits the specified data input into the system. Furthermore, GP simultaneously searches for and refines a model’s parameters and ultimately its structure.

The motivation for this paper is to investigate the use of GP algorithm and several neural networks techniques combined with ARMA models in order to improve the forecasting performance using autoregressive terms as inputs. This is achieved by comparing six benchmark neural network combined architectures with a forecast produced by the GP Algorithm. Most notably, classic neural networks as Multilayer Perceptron (MLP), Higher Order Neural Network (HONN), Recurrent Neural Network (RNN), autoregressive moving average model (ARMA), or technical models such as a moving average convergence/divergence model (MACD), plus a naïve trading strategy are all reviewed as benchmark methods.

From the analysis it emerges that the GP algorithm demonstrates a remarkable performance and outperforms all other models in a simple trading simulation exercise. This is also true when more sophisticated trading strategies are utilized with the application of confirmation filters and leverages as GP still demonstrates superior forecasting ability in terms of annualised return. It is worth mentioning the second best performance of the Hybrid HONNs and the Mixed HONNs. Dunis *et al.* (2010a, b) stress that the combination of neural networks can produce better forecasts compared with alternative techniques. Furthermore the Hybrid MLP and the Mixed MLP also perform well. Also, the RNNs which historically have performed remarkably well display less impressive forecasting potential in this research. It is observed that this might be due to the fact that they have an inability to provide accurate results when only autoregressive terms are used as inputs.

The remainder of the paper is organised as follows. In section 2, we present the literature relevant to GP modelling, the Hybrid, Mixed Neural Networks, the Recurrent Neural Network, the Higher Order Neural Networks and the Multilayer Percepton. Section 3 describes the dataset used for this research and its characteristics. An overview of the different neural network models, Genetic Programming algorithm and statistical techniques is given in section 4. Section 5 displays the empirical results of all the models considered and investigates the possibility of improving their performance with the application of more sophisticated trading strategies. Ultimately, Section 6 provides some concluding remarks.

2. LITERATURE REVIEW

The purpose of this investigation is to apply the GP algorithm to the ASE 20 Greek Stock market data comparing its results with the most promising new neural networks architectures combining them with autoregressive models (in our case the ARMA model) which have been developed recently with the purpose to overcome the numerous limitations of the more classic neural architectures and to assess whether they can achieve a higher performance in a trading simulation

GP was first developed by Barricelli (1954) as evolutionary algorithms. Progressively into the 1960's and 1970's these 'evolutionary algorithms' became more commonly known and recognized as optimization methods. In particular, Rechenberg (1971) and his research team were able to solve complex engineering problems through the application of optimization methods as documented in his 1971 PhD thesis. Furthermore Holland (1975) was another influential figure in the 1970's However, Fogel *et al*. (1964) are among the earliest practitioners pioneering in GP methodology. They apply evolutionary algorithms to the problem of discovering finite-state automata. In the development of GP methodology it was later adapted to the Markov decision making process. More importantly the first evidence of GP as the 'tree based' method that we are familiar with in modern financial forecasting was provided by Cramer (1985). More recently, Cramer's work has been expanded further by John R. Koza (1990), Koza (1992), Koza (1994), Koza (1998) and Koza *et al*. (1999, 2003) who apply these methodologies to complex optimization and search problems.

Although GP has now been established as a credible and respected technique this was not always the case. For example in the 1990's GP was considered incomprehensible. Enter the 2000's and the theory of GP has seen progressive and formidable growth. This has particularly been the case in the area of probabilistic models as GP has been incorporated with schema theories and Markov chain models. A variety of Genetic Programming applications is shown in the papers below: Wincler (2004), Wincler *et al.* (2004a, b), Madar *et al.* (2004, 2005), Willis *et al.* (1997), Tsang *et al.* (1998), Fukunaga and Stechert (1998) and Werner and Fogarty (2001).

On the other hand combining different models can increase the chance to capture different patterns in the data and improve forecasting performance. Several empirical studies have already suggested that by combining several different models, forecasting accuracy can often be improved over the individual model. Using hybrid models or combining several models has become a common practice to improve the forecasting accuracy since the well-known M-competition (Makridakis *et al.*(1982)) in which combination of forecasts from more than one model often led to improved forecasting performance. The basic idea of the model combination in forecasting is to use each model's unique feature to capture different patterns in the data. Both theoretical and empirical findings suggest that combining different methods can be an effective and efficient way to improve forecasts (Makridakis (1989), Newbold *et al*. (1974), Palm *et al*. (1992)). Research in time series forecasting argues that predictive performance improves in combined models. (Bishop

(1994), Clemen (1989), Hansen *et al*. (2003), Hibbert *et al*. (2000), Terui *et al*. (2002), Tseng *et al*. (2002), Zhang, (2003), Zhang *et al*. (2005).

RNNs have an activation feedback which embodies short-term memory allowing them to learn extremely complex temporal patterns. Their superiority against feedfoward networks when performing nonlinear time series prediction is well documented in Connor *et al.* (1993) and Adam *et al.* (1994). In financial applications, Kamijo *et al*. (1990) applied them successfully to the recognition of stock patterns of the Tokyo stock exchange while Tenti (1996) achieved remarkable results using RNNs to forecast the exchange rate of the Deutsche Mark. Tino *et al.* (2001) use them to trade successfully the volatility of the DAX and the FTSE 100 using straddles while Dunis and Huang (2002), using continuous implied volatility data from the currency options market, obtain remarkable results for their GBP/USD and USD/JPY exchange rate volatility trading simulation.

HONNs were first introduced by introduced by Giles and Maxwell (1987) as a fast learning network with increased learning capabilities. Although their function approximation superiority over the more traditional architectures is well documented in the literature (see among others Redding *et al*. (1993), Kosmatopoulos *et al*. (1995) and Psaltis *et al*. (1998)), their use in finance so far has been limited. This has changed when scientists started to investigate not only the benefits of Neural Networks (NNs) against the more traditional statistical techniques but also the differences between the different NNs model architectures. Practical applications have now verified the theoretical advantages of HONNs by demonstrating their superior forecasting ability and put them in the front line of research in financial forecasting. For example Dunis *et al.* (2006b) use them to forecast successfully the gasoline crack spread while Fultcher *et al.* (2006) apply HONNs to forecast the AUD/USD exchange rate, achieving a 90% accuracy. However, Dunis *et al.* (2006a) show that, in the case of the futures spreads and for the period under review, the MLPs performed better compared with HONNs and recurrent neural networks. Moreover, Dunis *et al.* (2008a), who also study the EUR/USD series for a period of 10 years, demonstrate that when multivariate series are used as inputs the HONNs, RNN and MLP networks have a similar forecasting power. Finally, Dunis *et al.* (2008b) in a paper with a methodology identical to that used in this research, demonstrate that HONN and the MLP networks are superior in forecasting the EUR/USD ECB fixing until the end of 2007, compared to the RNN networks, an ARMA model, a MACD and a naïve strategy.

3. THE ASE 20 GREEK INDEX AND RELATED FINANCIAL DATA

For Futures on the FTSE/ASE-20 that are traded in derivatives markets the underlying asset is the blue chip index FTSE/ASE-20. The FTSE/ASE-20 index is based on the 20 largest ASE stocks. It was developed in 1997 by the partnership of ASE with FTSE International and is the established benchmark.

It represents over 50% of ASE's total capitalisation and currently has a heavier weight on banking, telecommunication and energy stocks.

The FTSE/ASE 20 index is traded as a futures contract that is cash settled upon maturity of the contract with the value of the index fluctuating on a daily basis. The cash settlement of this index is simply determined by calculating the difference between the traded price and the closing price of the index on the expiration day of the contract. Furthermore, settlement is reached between each of the participating counterparties. Whilst the futures contract is traded in index points the monetary value of the contract is calculated by multiplying the futures price by the multiple of 5 euros per point. For example, a contract trading at 1,400 points is valued at 7,000 EUR.

As a result, our application is deemed more realistic and specific to the series that we investigate in this paper[1].

Name of Period	Trading Days	Beginning	End
Total Dataset	2087	21 January 2001	31 December 2008
Training Dataset	1719	29 January 2001	30 August 2007
Out- of- sample Dataset(Validation Set)	349	31 August /2007	31 December 2008

Table 1: The ASE 20 dataset

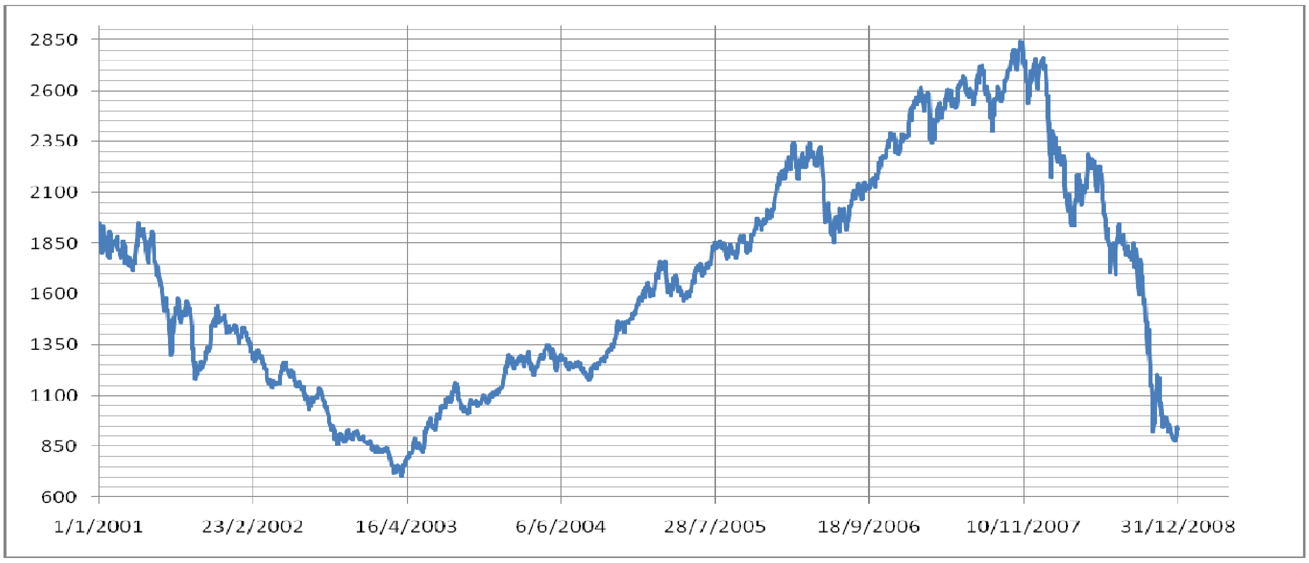


Fig. 1: ASE 20 fixing prices (total dataset).

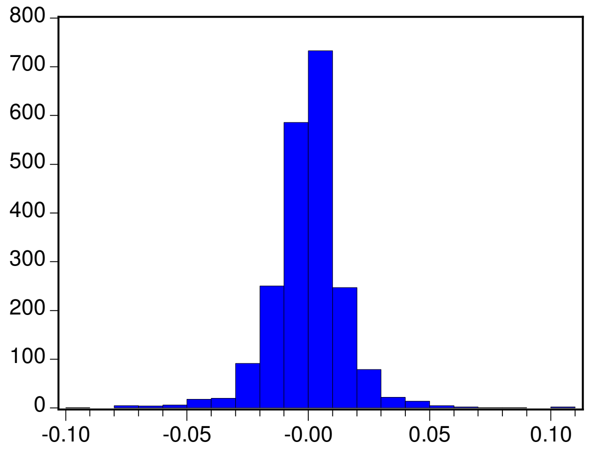
The observed ASE 20 time series is non-normal (Jarque-Bera statistics confirms this at the 99% confidence interval) containing slight skewness and high kurtosis. It is also non-stationary and we decided to transform the ASE 20 series into a stationary series of rates of return[2].

Given the price level P_1, P_2,…,P_t, the rate of return at time t is formed by:

[1] We examine the ASE 20 since its first trading day on 21 January 2001 (Greece's entrance in the European Monetary Zone), and until 31 December 2008, using the continuous data available from DataStream.

[2] The percentage return is linearly additive but the log return is not linearly additive across portfolio components.

$$R_t = \left(\frac{P_t}{P_{t-1}} \right) - 1 \qquad [1]$$



```
Series: RETURNS
Sample 1 2087
Observations 2087

Mean          -0.000240
Median         0.000000
Maximum        0.108214
Minimum       -0.093318
Std. Dev.      0.015088
Skewness      -0.036670
Kurtosis       9.514666

Jarque-Bera    3691.056
Probability    0.000000
```

Fig. 2: ASE 20 returns summary statistics (total dataset)

As inputs to our GP algorithm and our networks, based on the autocorrelation function and some ARMA experiments we selected 3 sets of autoregressive and moving average terms of the ASE 20 returns and the 1-day Risk Metrics volatility series.

Number	Variable	Lag
1	Athens Composite all share return	1
2	Athens Composite all share return	3
3	Athens Composite all share return	6
4	Athens Composite all share return	8
5	Athens Composite all share return	10
6	Athens Composite all share return	13
7	Athens Composite all share return	14
8	Moving Average of the Athens Composite all share return	15
9	Athens Composite all share return	16
10	Athens Composite all share return	18
11	Moving Average of the Athens Composite all share return	19

Table 2: Explanatory variables for traditional Neural Networks and the GP algorithm

Number	Variable	Lag
1	Athens Composite all share return	1
2	Athens Composite all share return	3
3	Athens Composite all share return	5
4	Athens Composite all share return	7
5	Athens Composite all share return	8
6	Athens Composite all share return	9
7	Athens Composite all share return	12
8	Athens Composite all share return	13
9	Moving Average of the Athens Composite all share return	14
10	Athens Composite all share return	15
11	Athens Composite all share return	16
12	Moving Average of the Athens Composite all share return	17
13	1-day Riskmetrics Volatility	1

Table 3: Explanatory variables for the hybrid neural networks

Number	Variable	Lag
1	Athens Composite all share return	1
2	Athens Composite all share return	2
3	Athens Composite all share return	4
4	Athens Composite all share return	5
5	Athens Composite all share return	7
6	Athens Composite all share return	9
7	Moving Average of the Athens Composite all share return	10
8	Athens Composite all share return	13
9	Athens Composite all share return	14
10	Athens Composite all share return	15
11	Moving Average of the Athens Composite all share return	16
12	Athens Composite all share return	17

Table 4: Explanatory variables for the mixed neural networks

In order to train the neural networks and the GP algorithm we further divided our dataset as follows:

Name of Period	Trading Days	Beginning	End
Total Dataset	2087	21 January 2001	31 December 2008
Training Dataset	1373	29 January 2001	03 May2006
Test Dataset	346	04 May 2006	30 August 2007
Out-of- sample Dataset (Validation Set)	349	31 August 2007	31 December 2008

Table 5: The neural networks and GP algorithm datasets

4. FORECASTING MODELS

4.1 Benchmark Models

In this paper, we benchmark our neural network models with 3 traditional strategies, namely an autoregressive moving average model (ARMA), a moving average convergence/divergence technical model (MACD) and a naïve strategy.

4.1.1 Naïve strategy

The naïve strategy simply takes the most recent period change as the best prediction of the future change. The model is defined by:

$$\hat{Y}_{t+1} = Y_t \quad [2]$$

Where Y_t is the actual rate of return at period *t*

$\hat{Y}_{t+1}$ is the forecast rate of return for the next period

The performance of the strategy is evaluated in terms of trading performance via a simulated trading strategy.

4.1.2 Moving Average

The moving average model is defined as:

$$M_t = \frac{(Y_t + Y_{t-1} + Y_{t-2} + ... + Y_{t-n+1})}{n} \qquad [3]$$

Where	M_t	is the moving average at time *t*
	n	is the number of terms in the moving average
	Y_t	is the actual rate of return at period *t*

The MACD strategy used is quite simple. Two moving average series are created with different moving average lengths. The decision rule for taking positions in the market is straightforward: If the short-term moving average intersects the long-term moving average from below a 'long' position is taken. Conversely, if the long-term moving average is intersected from above a 'short' position is taken[3].

The forecaster must use judgement when determining the number of periods *n* on which to base the moving averages. The combination that performed best over the in-sample sub-period was retained for out-of-sample evaluation. The model selected was a combination of the ASE 20 and its 7-day moving average, namely *n* = 1 and 7 respectively or a (1, 7) combination. The performance of this strategy is evaluated solely in terms of trading performance.

4.1.3 ARMA Model

Autoregressive moving average models (ARMA) assume that the value of a time series depends on its previous values (the autoregressive component) and on previous residual values (the moving average component)[4].

The ARMA model takes the form:

$$Y_t = \phi_0 + \phi_1 Y_{t-1} + \phi_2 Y_{t-2} + ... + \phi_p Y_{t-p} + \varepsilon_t - w_1 \varepsilon_{t-1} - w_2 \varepsilon_{t-2} - ... - w_q \varepsilon_{t-q} \qquad [4]$$

where	Y_t	is the dependent variable at time *t*
	Y_{t-1}, Y_{t-2}, and Y_{t-p}	are the lagged dependent variable
	ϕ_0, ϕ_1, ϕ_2, and ϕ_p	are regression coefficients
	ε_t	is the residual term
	ε_{t-1}, ε_{t-2}, and ε_{t-p}	are previous values of the residual
	w_1, w_2, and w_q	are weights.

Using as a guide the correlogram in the training and the test sub periods we have chosen a restricted ARMA (7, 7) model. All of its coefficients are significant at the 99% confidence interval. The null hypothesis that all coefficients (except the constant) are not significantly different from zero is rejected at the 99% confidence interval (see Appendix A1).

The selected ARMA model takes the form:

[3]A 'long' ASE 20 position means buying the index at the current price, while a 'short' position means selling the index at the current price.

[4] For a full discussion on the procedure, refer to Box *et al.* (1994) or Pindyck and Rubinfeld (1998).

$$Y_t = 2.90 \cdot 10^{-4} + 0.376Y_{t-1} - 0.245Y_{t-3} - 0.679Y_{t-7} + 0.374\varepsilon_{t-1} - 0.270\varepsilon_{t-3} - 0.677\varepsilon_{t-7} \quad [6]$$

The model selected was retained for out-of-sample estimation. The performance of the strategy is evaluated in terms of traditional forecasting accuracy and in terms of trading performance[5].

4.2 Various Neural Network Architectures

Neural networks exist in several forms in the literature. The most popular architecture is the Multi-Layer Perceptron (MLP).
A standard neural network has at least three layers. The first layer is called the input layer (the number of its nodes corresponds to the number of explanatory variables). The last layer is called the output layer (the number of its nodes corresponds to the number of response variables). An intermediary layer of nodes, the hidden layer, separates the input from the output layer. Its number of nodes defines the amount of complexity the model is capable of fitting. In addition, the input and hidden layer contain an extra node, called the bias node. This node has a fixed value of one and has the same function as the intercept in traditional regression models. Normally, each node of one layer has connections to all the other nodes of the next layer.

The network processes information as follows: the input nodes contain the value of the explanatory variables. Since each node connection represents a weight factor, the information reaches a single hidden layer node as the weighted sum of its inputs. Each node of the hidden layer passes the information through a nonlinear activation function and passes it on to the output layer if the calculated value is above a threshold.

The training of the network (which is the adjustment of its weights in the way that the network maps the input value of the training data to the corresponding output value) starts with randomly chosen weights and proceeds by applying a learning algorithm called backpropagation of errors[6] (Shapiro (2000)). The learning algorithm simply tries to find those weights which minimize an error function (normally the sum of all squared differences between target and actual values). Since networks with sufficient hidden nodes are able to learn the training data (as well as their outliers and their noise) by heart, it is crucial to stop the training procedure at the right time to prevent overfitting (this is called 'early stopping'). This can be achieved by dividing the dataset into 3 subsets respectively called the training and test sets used for simulating the data currently available to fit and tune the model and the validation set used for simulating future values. The network parameters are then estimated by fitting the training data using the above mentioned iterative procedure (backpropagation of errors). The iteration length is optimised by maximising the forecasting accuracy for the test dataset. Our networks, which are specially designed for financial purposes, will stop training when the profit of our forecasts in the test sub-period is

[5] Statistical measures are given in section 4.2.5 below.
[6]Backpropagation networks are the most common multi-layer networks and are the most commonly used type in financial time series forecasting (Kaastra and Boyd (1996)).

maximized. Then the predictive value of the model is evaluated applying it to the validation dataset (out-of-sample dataset).

There is a range of combination techniques that can be applied to forecasting the attempt to overcome some deficiencies of single models. The combining method aims at reducing the risk of using an inappropriate model by combining several to reduce the risk of failure. Typically this is done because the underlying process cannot easily be determined (Hibon and Evgeniou (2005)).

Combining methods involves using several redundant models designed for the same function, where the diversity of the components is thought important (Brown *et al.* 2005). The procedure of making a hybrid or a mixed forecasting time series model can be achieved by combining an ARMA process in order to learn the linear component of the conditional mean pattern through an artificial neural network process designed to learn its nonlinear elements. In summary, the proposed methodologies of the hybrid and mixed system will be explained in the next section in figures 6 and 7.

4.2.1 The Multi-Layer Perceptron Model Architecture

The network architecture of a 'standard' MLP looks as presented in figure 4[7]:
0251658240251659264

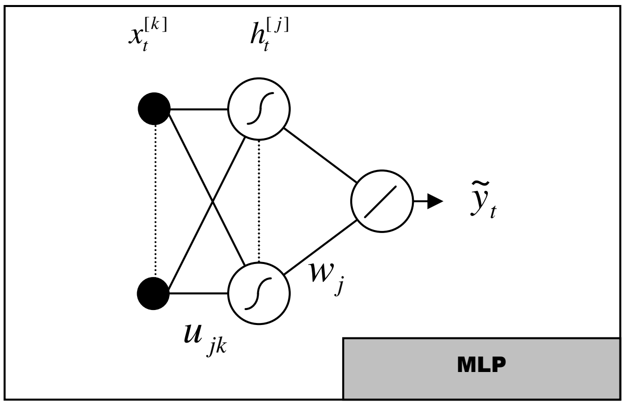


Fig. 4: A single output, fully connected MLP model

Where:

$x_t^{[n]}\ (n = 1,2,\cdots,k+1)$ are the model inputs (including the input bias node) at time t

$h_t^{[m]}\ (m = 1,2,...,j+1)$ are the hidden nodes outputs (including the hidden bias node)

$\tilde{y}_t$ is the MLP model output

[7] The bias nodes are not shown here for the sake of simplicity.

u_{jk} and w_j are the network weights

0251663360 is the transfer sigmoid function: $S(x)=\frac{1}{1+e^{-x}}$, [6]

is a linear function: $F(x)=\sum_i x_i$ [7]

The error function to be minimised is:

$E(u_{jk},w_j)=\frac{1}{T}\sum_{t=1}^{T}\left(y_t-\tilde{y}_t(u_{jk},w_j)\right)^2$, with y_t being the target value [8

4.2.2 The Recurrent Network Architecture

Our next model is the recurrent neural network. While a comple
of RNN models is beyond the scope of this paper, we prese
explanation of the significant differences between
architectures. For an exact specification of the recurrent
(1990).

A simple recurrent network has activation feedbac
term memory. The advantages of using recurrent
networks, for modelling non-linear time series,
the past. However as described in Tenti (19
RNNs is that they require substantially more
in simulation, than standard backpropaga
substantial increase in computational ti
can yield better results in comparison
memory inputs.

A simple illustration of the archite

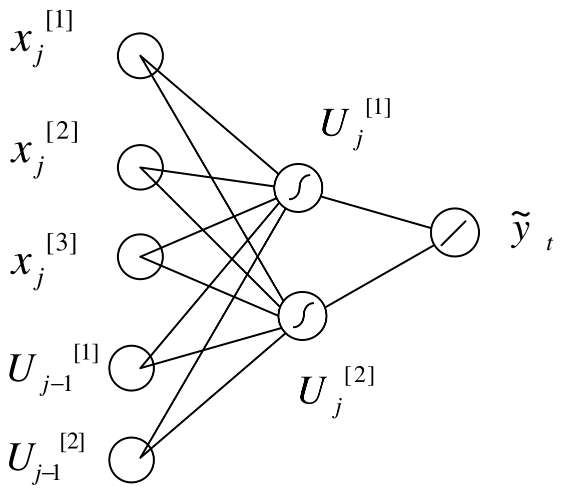


Fig. 5: Elman recurrent neural network architecture with two nodes on the hidden layer

Where:

$x_t^{[n]}\ (n = 1,2,\cdots,k+1)$, $u_t^{[1]}, u_t^{[2]}$ are the model inputs (including the input bias node) at time *t*

$\tilde{y}_t$ is the recurrent model output

$d_t^{[f]}\ (f = 1,2)$ and $w_t^{[n]}\ (n = 1,2,\cdots,k+1)$ are the network weights

$U_t^{[f]}\ (f = 1,2)$ is the output of the hidden nodes at time t

0∫666432 is the transfer sigmoid function: $S(x) = \dfrac{1}{1+e^{-x}}$, [9]

⊘ is the linear output function: $F(x) = \sum_i x_i$ [10]

The error function to be minimised is:

$$E(d_t, w_t) = \frac{1}{T}\sum_{t=1}^{T}\left(y_t - \tilde{y}_t(d_t, w_t)\right)^2$$

In short, the RNN architecture can provide more accurate ou the inputs are (potentially) taken from all previous values (and $U_{j-1}^{[2]}$ in the figure above).

4.2.3 The Higher Order Neural Network Architec

Higher Order Neural Networks (HONNs) were Maxwell (1987) and were called "Tensor Ne their use in finance has so far been limited show that, with shorter computational times and limited input variables, "the best HONN models show a profit increase over the MLP of around 8%" on the EUR/USD time series (p. 7). For Zhang *et al.* (2002), a significant advantage of HONNs is that "HONN models are able to provide some rationale for the simulations they produce and thus can be regarded as "open box" rather than "black box". HONNs are able to simulate higher frequency, higher order non-linear data, and consequently provide superior simulations compared to those produced by ANNs (Artificial Neural Networks)" (p. 188). Furthermore HONNs clearly outperform in terms of annualised return and this enables Dunis *et al.* (2008a) to conclude with confidence over their forecasting superiority and their stability and robustness through time.

While they have already experienced some success in the field of pattern recognition and associative recall[8], HONNs have only started recently to be used in finance. The architecture of a three input second order HONN is shown below:

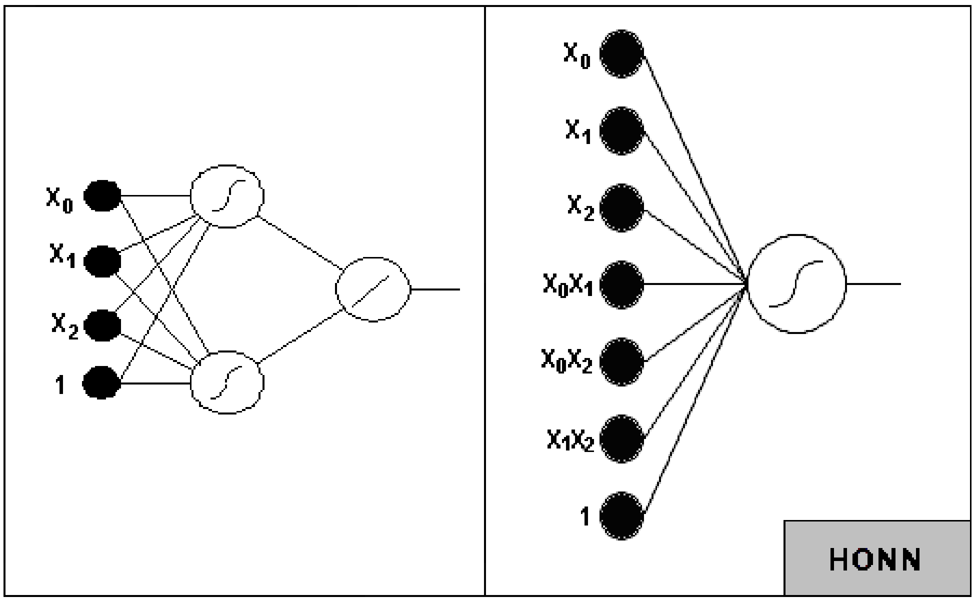


Fig. 6: Left, MLP with three inputs and two hidden nodes; right, second order HONN with three inputs

Where:

$x_t^{[n]}\ (n = 1,2,\cdots,k+1)$ are the model inputs (including the input bias node) at time t

$\tilde{y}_t$ is the HONNs model output

u_{jk} are the network weights

● are the model inputs.

[8] Associative recall is the act of associating two seemingly unrelated entities, such as smell and colour. For more information see Karayiannis *et al.* (1994).

02 669504 is the transfer sigmoid function: $S(x) = \frac{1}{1+e^{-x}}$, [12]

is a linear function: $F(x) = \sum_i x_i$ [13]

The error function to be minimised is:

$E(u_{jk}, w_j) = \frac{1}{T}\sum_{t=1}^{T}(y_t - \tilde{y}_t(u_{jk},))^2$, with y_t being the target value [14]

HONNs use joint activation functions; this technique reduces the need to establish the relationships between inputs when training. Furthermore th reduces the number of free weights and means that HONNS are faster to than even MLPs. However because the number of inputs can be very la higher order architectures, orders of 4 and over are rarely used.

Another advantage of the reduction of free weights means that t of overfitting and local optima affecting the results of neural ne largely avoided. For a complete description of HONNs se (2005- page 52).

4.2.4 THE HYBRID HONN, MLP AND RNN ARCHITE

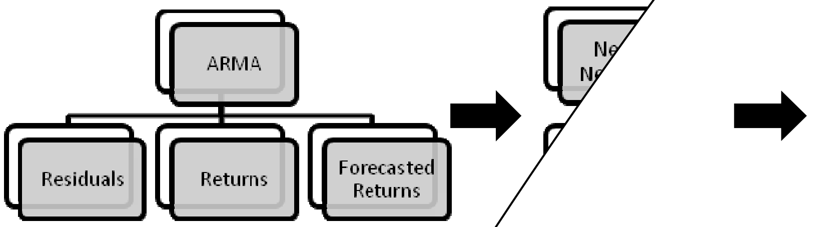
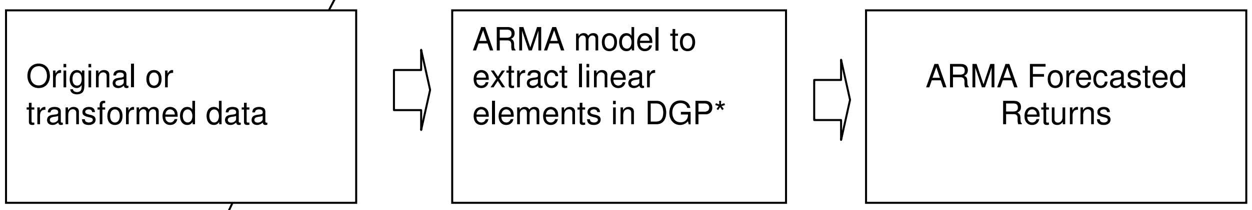
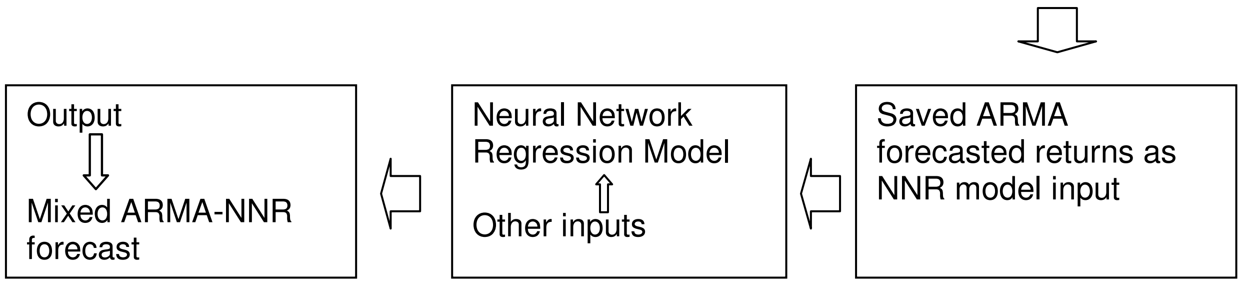


Fig. 7: The architecture o

The methodology we follow into 3 steps. In the first st second step we forecast step we create the hy ARMA model with the

4.2.5 THE MIXED


Original or transformed data
ARMA model to extract linear elements in DGP*
ARMA Forecasted Returns


Output
Mixed ARMA-NNR forecast
Neural Network Regression Model
Other inputs
Saved ARMA forecasted returns as NNR model input

*DGP= Data Generating Process

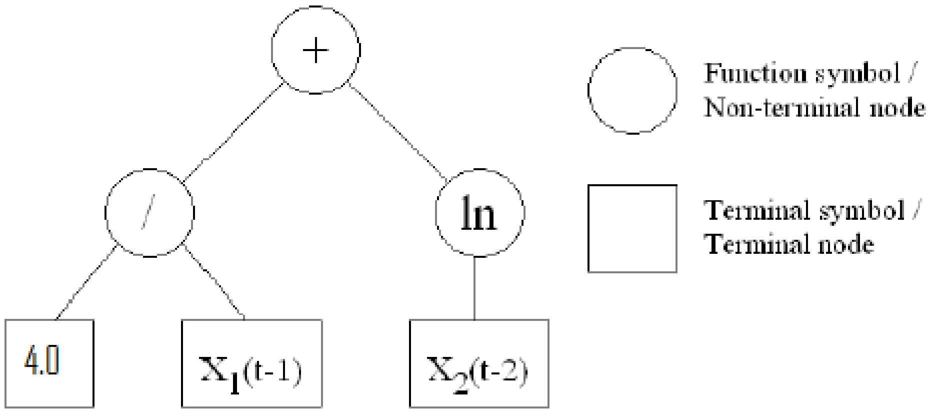
Fig. 8: The architecture of a Mixed Neural Network Model

The methodology we follow to construct the mixed ARMA-NNR model is divided into 2 steps. In the first step the ASE 20 index is modelled with a traditional ARMA model. In the second step the forecasted returns of the ARMA model are used as an input to the neural network for forecasting the selected time series.

4.3 The Genetic Programming Algorithm

For the purpose of our research, the GP application is coded and implemented to evolve tree based structures that present models (sub trees) of input – output. In the design phase of our GP application we focused primarily on execution time optimization as well as limiting the 'bloat effect'. The bloat effect is similar to the issue of overfitting experienced in Neural Networks however in our case we run a risk of continuously increasing and expanding the tree size. This algorithm is run in a steady state in that a single member of the population is replaced at a time. Furthermore, our GP application reproduces newer models replacing the weaker ones in the population according to their fitness. Reasoning behind the decision to use a steady state algorithm is justified as they hold a greater selection strength and genetic drift over other algorithms such as a typical generational GAs. Additionally, steady state algorithms also offer exceptional multiprocessing capabilities.

In our application of the genetic programming we utilize formulas to evolve algebraic expressions that enable the analysis / optimization of results in a 'tree like structure'. This genetic tree structure consists of nodes (depicted as circles in the diagram below) which are essentially functions that perform actions within this structure. Furthermore, these functions are in place to generate output signals. On the other hand, the squares in the tree signify terminal functions representing the end of a function once the most superior sub tree (model) is achieved. For example, the below tree structure (model) is characterized by the algebraic expression $4.0/x_1$ (t-1) + ln(x2(t-2)). In this case there is one output and the terminal nodes are constant at 4. Additionally, the outputs are expressed by $x_1(t-1)$ and $x_2(t-2)$. In the execution of the genetic algorithm it has to be understood that each individual in the population correspond to a single sub tree structure. Each of these sub trees are limited by the predefined maximum tree size set to 6 in our application.



+
/
ln
4.0
X1(t-1)
X2(t-2)
Function symbol / Non-terminal node
Terminal symbol / Terminal node

Fig. 9, Example of a tree structure

Koza (1998) summarises the functionality aspect of the GP algorithm in the following steps:

(1) The generation of an initial population of randomly constructed models is developed with each model being represented in a tree like structure as discussed previously. Additionally, the evolutionary algorithm represents each chromosome of the population as a tree of variable length (i.e. total number of functions and terminals) or a maximum depth of the model tree. The process of randomly reproducing each variable of the population is completed once all of these functions of the tree are terminal symbols. However, until the process is halted by these 'terminal symbols' then the tree like structure of chromosomes continues to multiply (grow) with each generation as the population expands to not only include the parents but also their offspring. This is achieved by crossover and mutation operators. On the whole, it also has to be understood that the majority of these models produced in the initial population are, in most cases, unsatisfactory when tested for their performance with some individual models 'fitting' better than others. However, one of the virtues offered by Genetic Programming is that they exploit and manipulate these differences until the best fitting models, in terms of least error, are produced.

(2) Following this initial generation of randomly selected models a random subset (sub tree) of the population is then selected for a tournament. Hence this process is known as a tournament selection phase. This process (tournament procedure) is essentially a selection mechanism to decipher which individuals from the population are to be selected for reproduction to develop the next generation.

(3) An evaluation of the members of this subset is then carried out and assigned a fitness value. As stated by Koza (1998) the fitness cases are either selected at random or in some structured manner (e.g. at regular intervals). In our application, as mentioned briefly in the first step, the fitness value is defined as the mean squared error (MSE) with the lowest MSE being targeted as the best. Furthermore, the fitness may be measured in terms of the sum of the absolute value of the differences between the output produced by the model and/or the desired output (i.e. the Minkowski distance) or, alternatively, the square root of the sum of the squared errors (i.e. the Euclidean distance).

(4) Following the establishment of fitness values the tournament winners are then determined. To reiterate, the winners of this scenario are the models with the lower MSE.

(5) Having identified the tournament winners in the previous step we then proceed by exposing the models to two genetic operators known as mutations and crossovers. Both operators are discussed in more detail below:

Mutation: This is the creation of a new model that is mutated randomly from an existing one as circled in the diagram below (1*). This one mutation point is indiscriminately chosen as an independent point and the resulting sub-tree is to be omitted. From this resulting sub-tree, another new sub-tree (2*) is then reproduced using the same procedure that was initially implemented to create the original random population. Although this was the procedure we implemented for mutation there are also a number of alternative methods that are explored in other research.

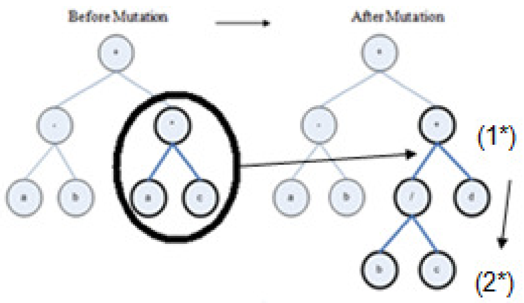


Fig 10, Mutation tree structure example

Crossover: This operator creates two new models from existing models by genetically recombining randomly chosen parts of them. This is achieved by using the crossover operation applied at a randomly chosen crossover point within each model. Due to the fact that entire sub-trees are swapped (from point 1* to point 2* and from points 3* to 4*), the crossover operation produces models as offsprings. Furthermore, the models are selected based on their fitness and the crossover allocates future trials to regions of the search space whose models contain parts from superior models. As a full explanation of crossovers is beyond the scope of this paper please refer to Koza (1992) for more details.

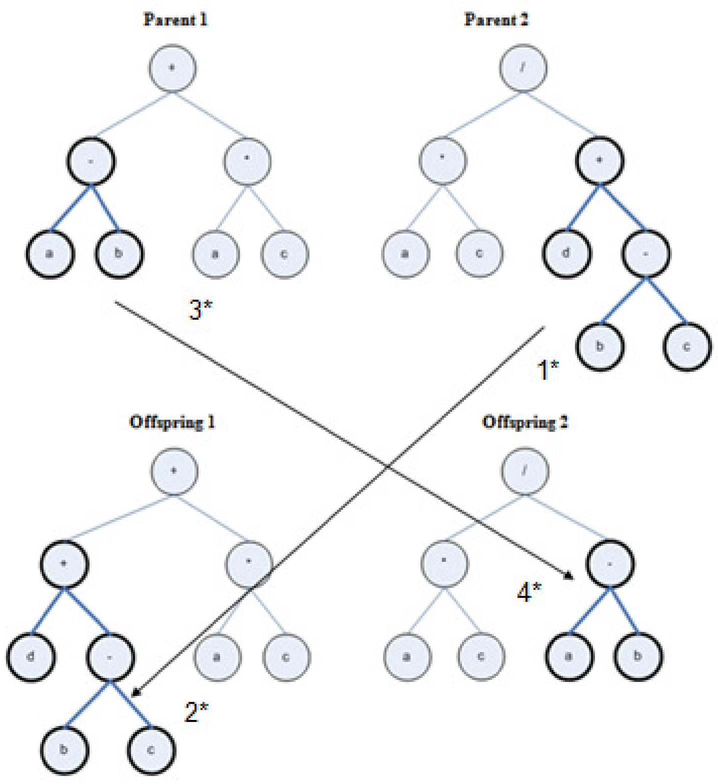
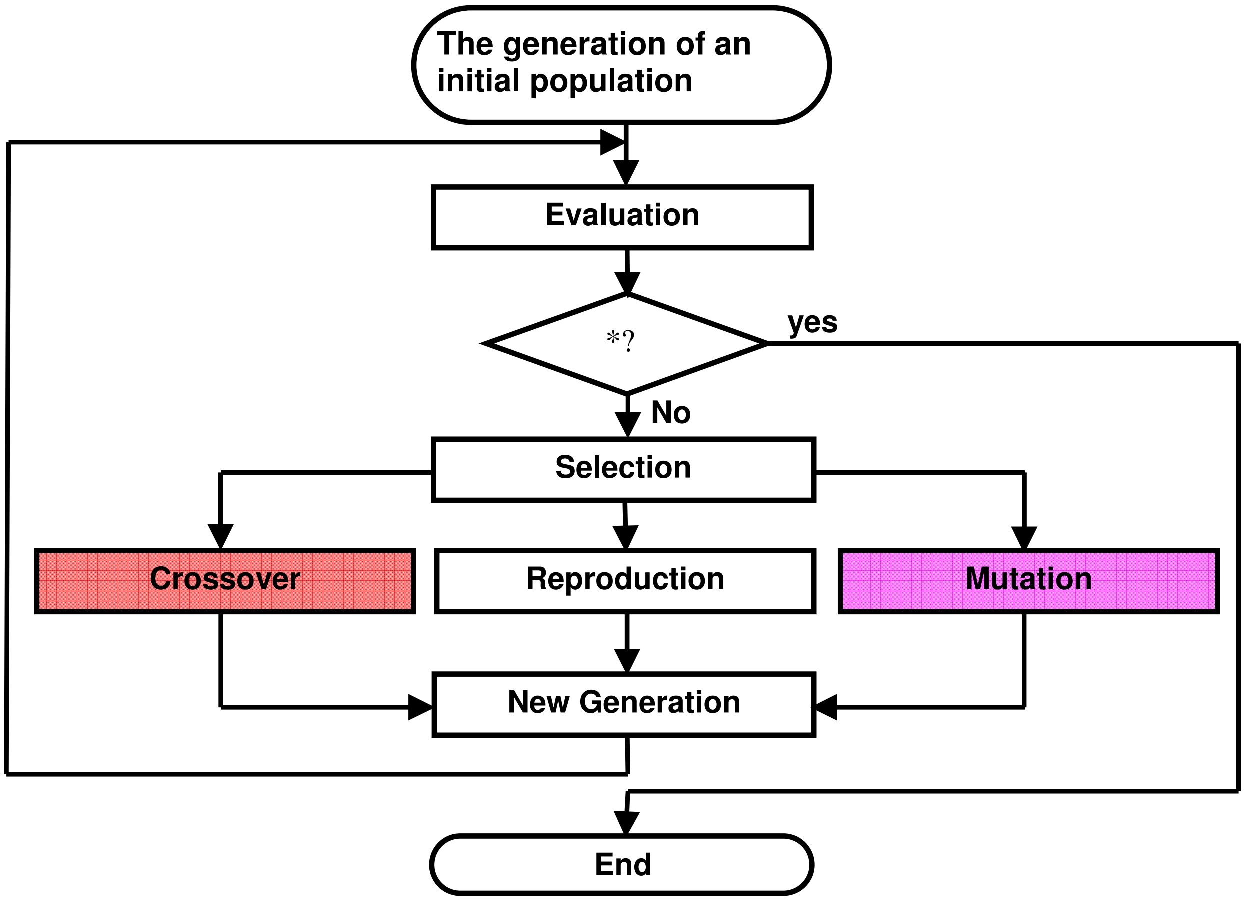


Fig. 11 , Crossover family tree like structure example

(6) The population is then altered with the tournament losers being replaced by the winners (superior) offspring.
(7) Provided the termination criterion (depicted as the symbol '?' in the following flow of stages) is not reached, the algorithm returns to step 2 and these steps are repeated until the predefined termination criterion for genetic programming is satisfied. In our study we have set the termination criterion to 100,000 at which point the cycles are stopped and forecasted results can be obtained.
(8) Ultimately, this protocol produces the best individual (model) of the population as a result.



*note: the symbol '?' is the termination criterion which iterates or terminates the procedure of GP.

Fig. 12: The architecture of Genetic Programming Algorithm

4.4 Settings for Genetic Programming Parameters (See Appendix A.4)

The parameters used for the optimization of our individual models are defined in order to yield better results and are specified as follows:

1. Population size 200. The population size is the total number of randomly chosen models in our experiment. This number can be altered however in our specific case we found that it was more beneficial (in terms of annualised returns) to set the population to 200 individuals. Each individual model has a tree structure composed of a set of functions and terminals. In summary, every model is a mathematical equation which participates in the program until the GP produces the best individual program.

2. Maximum tree depth 6. The maximum tree depth is the maximum length of each model (of each tree structure). In neural networks this is commonly known as hidden nodes. The depth depends on the functions and terminals of each individual model.
3. Tournament size 4. Tournament size is the size of models in the subset. Through trial and error we found this to be the most appropriate size.
4. Crossover trial 1. Crossover trial means the number of generations that we let the genetic programming algorithm to run. Crossover is achieved by creating two new offspring models for the new population by randomly recombining parts from two selected parents. In this experiment we have one crossover trial per generation.
5. Mutation probability 0.75. The mutation probability is the probability that can mutate parts of individual models from an existing one. Specifically mutation is performed by randomly selecting a parent with a probability related to its fitness, after that mutation randomly changes one or more genes representing part of the solution it encodes. Due to the fact that the population is 200 models, we use a relatively large probability. The mutation probability extends from an initial 0.1 and finishes at 0.9.

5 EMPIRICAL TRADING SIMULATION RESULTS

The trading performance of all the models considered in the validation subset is presented in the table below. We choose the network with the higher profit in the test sub-period. Our trading strategy applied is simple and identical for all the models: go or stay long when the forecast return is above zero and go or stay short when the forecast return is below zero**.** Appendix A.3 provides the performance of all the NNs and the GP Algorithm in the training and the test sub-periods while Appendix A.5, A.4 and A.2 provide the characteristics of our models and the performance measures. The Hybrid-RNNs, Mixed-RNNs are trained with gradient descent as for the Hybrid-MLPs and Mixed-MLPs. However, the increase in the number of weights, as mentioned before, makes the training process extremely slow: to derive our results, we needed about ten times the time needed with the Hybrid-MLPs and Mixed-MLPs. As shown in table 6 below, the Mixed-RNN, Hybrid-RNN have a slightly lower performance compared to the Hybrid-MLP, Hybrid-HONN, Mixed-MLP, Mixed-HONN and GP algorithm.

	NAIVE	MACD	ARMA	MLP	RNN	HONN
Information Ratio (excluding costs)	0.32	0.46	0.20	0.60	0.59	0.70
Annualised Volatility (excluding costs)	36.70%	38.12%	38.13%	38.11%	38.11%	38.10%
Annualised Return (excluding costs)	11.42%	17.63%	7.68%	22.99%	22.51%	26.75%
Maximum Drawdown	-49.41%	-50.63%	-36.50%	-36.26%	-36.22%	-38.71%

(excluding costs)						
Positions Taken (annualised)	119	38	72	105	147	98

	Hybrid MLP	Hybrid RNN	Hybrid*HONN*
Information Ratio (excluding costs)	0.86	0.81	0.94
Annualised Volatility (excluding costs)	38.08%	38.09%	38.07%
Annualised Return (excluding costs)	32.80%	30.72%	35.67%
Maximum Drawdown (excluding costs)	-59.05%	-59.05%	-59.05%
Positions Taken (annualised)	94	93	94

	Mixed MLP	Mixed RNN	Mixed HONN
Information Ratio (excluding costs)	0.83	0.78	0.91
Annualised Volatility (excluding costs)	38.08%	38.09%	38.07%
Annualised Return (excluding costs)	31.79%	29.63%	34.75%
Maximum Drawdown (excluding costs)	-26.29%	-27.94%	-28.20%
Positions Taken (annualised)	41	57	65

	GP Algorithm
Information Ratio (excluding costs)	1.03
Annualised Volatility (excluding costs)	38.04%
Annualised Return (excluding costs)	39.33%
Maximum Drawdown (excluding costs)	-28.20%
Positions Taken (annualised)	67

Table 6: Trading performance results

We can see that the GP algorithm performs significantly better than the Hybrid-HONNs, Hybrid-MLPs, Mixed HONNs, Mixed MLPs Hybrid-RNNs, and the Mixed-RNNs with similar sorts of drawdowns, and significantly better than the standard neural network architectures.

Up to now, we have presented the trading results of all our models without considering transaction costs. Since some of our models trade quite often, taking transaction costs into account might change the whole picture. Following Dunis *et al.* (2008a), we check for potential improvements to our models through the application of confirmation filters. Confirmation filters are trading strategies devised to filter out those trades with expected returns below a threshold d around zero. They suggest to go long when the forecast is above d and to go short when the forecast is below d. It just so happens that the Mixed ARMA-Neural Network models perform best without any filter. This is also the case of the MLP and HONN models. Still, the application of confirmation filters to the benchmark models and the RNN model could have led to these models outperforming the Mixed, MLP HONN models. This is not the case however but, in order to conserve space, these results are not shown here but they are available from the authors.

5.1 Transaction Costs

According to the Athens Stock Exchange, transaction costs for financial institutions and fund managers dealing a minimum of 143 contracts or 1 million Euros is 10 Euros per contract (round trip). Dividing this transaction cost of the 143 contracts by average size deal (1 million Euros) gives us an average transaction cost for large players of 14 basis points (1 basis point=1/100 of 1%) or 0.14% per position.

	NAIVE	MACD	ARMA	MLP	RNN	HONN
Information Ratio (excluding costs)	0.32	0.46	0.20	0.60	0.59	0.70
Annualised Volatility (excluding costs)	36.70%	38.12%	38.13%	38.11%	38.11%	38.10%
Annualised Return (excluding costs)	11.42%	17.63%	7.68%	22.99%	22.51%	26.75%
Maximum Drawdown (excluding costs)	-49.41%	-50.63%	-36.50%	-36.26%	-36.22%	-38.71%
Positions Taken (annualised)	119	38	72	105	147	98
Transaction costs	15.47%	4.94%	9.36%	13.65%	19.11%	12.74%
Annualised Return (including costs)	-4.05%	12.69%	-1.68%	9.35%	3.40%	14.01%

		Hybrid MLP	Hybrid RNN	Hybrid HONN
Information Ratio	***(excluding costs)***	0.86	0.81	0.94
Annualised Volatility	***(excluding costs)***	38.08%	38.09%	38.07%
Annualised Return	***(excluding costs)***	32.80%	30.72%	35.67%
Maximum Drawdown	***(excluding costs)***	-59.05%	-59.05%	-59.05%
Positions Taken	***(annualised)***	94	93	94
Transaction costs		12.22%	12.09%	12.22%
Annualised Return	***(including costs)***	20.58%	18.63%	23.45%

	Mixed MLP	Mixed RNN	Mixed HONN
Information Ratio (excluding costs)	0.83	0.78	0.91
Annualised Volatility (excluding costs)	38.08%	38.09%	38.07%
Annualised Return (excluding costs)	31.79%	29.63%	34.75%
Maximum Drawdown (excluding costs)	-26.29%	-27.94%	-28.20%
Positions Taken (annualised)	41	57	65
Transaction costs	5.74%	7.98%	9.10%

Annualised Return (including costs)	26.05%	21.65%	25.65%

	GP Algorithm
Information Ratio (excluding costs)	1.03
Annualised Volatility (excluding costs)	38.04%
Annualised Return (excluding costs)	39.33%
Maximum Drawdown (excluding costs)	-28.20%
Positions Taken (annualised)	67
Transaction costs	9.40%
Annualised Return (including costs)	29.93%

Table 7: Out-of-sample results with transaction costs

We can see that, after transaction costs, the GP algorithm model outperforms all the other strategies based on the annualized net return closely followed by the Mixed-MLP, the Mixed HONN and the Hybrid HONNs strategy. On the other hand, the naïve strategy and the ARMA model produce negative results after transaction costs are taken into account. The HONN and MACD achieve decent returns, yet well below those produced by our best models.

5.2 Leverage to Exploit High Sharpe Ratios

In order to further improve the trading performance of our models we introduce a "level of confidence" to our forecasts, i.e. a leverage based on the test sub-period. For the naïve model, which presents a negative return we do not apply leverage. The leverage factors applied are calculated in such a way that each model has a common volatility of 20%[9] on the test data set.

The transaction costs are calculated by taking 0.14% per position into account, while the cost of leverage (interest payments for the additional capital) is calculated at 4% p.a. (that is 0.016% per trading day[10]). Our final results are presented in table 8 below.

	NAIVE	**MACD**	**ARMA**	**MLP**	**RNN**	**HONN**
Information Ratio (excluding costs)	0.32	0.70	0.20	0.60	0.59	0.70
Annualised Volatility (excluding costs)	36.70%	40.03%	38.13%	40.28%	40.21%	40.31%

[9] Since most of the models have a volatility of about 20%, we have chosen this level as our basis. The leverage factors retained are given in table 8.

[10] The interest costs are calculated by considering a 4% interest rate p.a. divided by 252 trading days. In reality, leverage costs also apply during non-trading days so that we should calculate the interest costs using 360 days per year. But for the sake of simplicity, we use the approximation of 252 trading days to spread the leverage costs of non-trading days equally over the trading days. This approximation prevents us from keeping track of how many non-trading days we hold a position.

Annualised Return (excluding costs)	11.42%	18.51%	7.68%	24.30%	23.75%	28.30%
Maximum Drawdown (excluding costs)	-49.41%	-53.16%	-36.50%	-38.32%	-38.21%	-40.96%
Leverage Factor	-	1.050	-	1.057	1.055	1.058
Positions Taken (annualised)	119	38	72	105	147	98
Transaction and leverage costs	15.47%	4.94%	9.36%	13.65%	19.11%	12.74%
Annualised Return (including costs)	-4.05%	13.57%	-1.68%	10.65%	4.64%	15.56%

	Hybrid-MLP	**Hybrid-RNN**	**Hybrid-HONN**
Information Ratio (excluding costs)	0.86	0.81	0.94
Annualised Volatility (excluding costs)	40.14%	40.30%	40.24%
Annualised Return (excluding costs)	34.57%	32.50%	37.71%
Maximum Drawdown (excluding costs)	-62.24%	-62.48%	-62.46%
Leverage Factor	1.054	1.058	1.057
Positions Taken (annualised)	94	93	94
Transaction and leverage costs	12.22%	12.1%	12.22%
Annualised Return (including costs)	12.35%	20.4%	24.89

	Mixed-MLP	**Mixed-RNN**	**Mixed-HONN**
Information Ratio (excluding costs)	0.83	0.78	0.91
Annualised Volatility (excluding costs)	40.22%	40.22%	40.17%
Annualised Return (excluding costs)	33.57%	31.29%	36.67%
Maximum Drawdown (excluding costs)	-27.76%	-29.50%	-29.75%
Leverage Factor	1.056	1.056	1.055
Positions Taken (annualised)	41	57	65
Transaction and leverage costs	6.052%	8.30%	9.40%
Annualised Return (including costs)	27.51%	23.00%	27.27%

	GP Algorithm
Information Ratio (excluding costs)	1.03
Annualised Volatility (excluding costs)	41.84%
Annualised Return (excluding costs)	43.26%
Maximum Drawdown (excluding costs)	-31.02%
Leverage Factor	1.10

Positions Taken (annualised)	67
Transaction and leverage costs	9.95%
Annualised Return (including costs)	33.34%

Table 8: Trading performance - final results

As can be seen from table 8, the GP algorithm continues to demonstrate a superior trading performance despite significant drawdowns. The Mixed HONN, the Mixed MLP and the Hybrid HONN strategies also perform well and presents high annualised returns. In general, we observe that all models are able to gain extra profits from the leverage as the increased transaction costs countered by increased performance Again it is worth mentioning, that the time needed to train the HONN, the Hybrid-HONN and the Mixed-HONN network was considerably shorter compared with that needed for the MLP, Hybrid-MLP, Mixed-MLP, RNN, Mixed-RNN and the Hybrid-RNN networks.

6. CONCLUDING REMARKS

In this paper, we apply a Genetic Programming algorithm, Multi-layer Perceptron, Recurrent, Higher Order, Mixed-Multilayer Perceptron, Mixed-Recurrent, Mixed-Higher Order neural networks, Hybrid-Multilayer Perceptron, Hybrid-Recurrent, Hybrid-Higher Order neural networks to a one-day-ahead forecasting and trading task of the ASE 20 fixing series with only autoregressive terms as inputs. We use a naïve strategy, a MACD and an ARMA model as benchmarks. We develop these different prediction models over the period January 2001 - August 2007 and validate their out-of-sample trading efficiency over the following period from September 2007 through December 2008.

The GP algorithm demonstrates a higher trading performance in terms of annualised return and information ratio before transaction costs. When more elaborate trading strategies are applied and transaction costs are considered the GP algorithm again continues to outperform all other models achieving the highest annualised return. The Mixed-HONNs, the Mixed-RNNs and the Hybrid-HONNs models perform remarkably as well and seem to have ability in providing good forecasts when autoregressive series are only used as inputs.

It is also important to note that the Mixed-MLP network which presents a very close second best performance needs less training time than the GP algorithm, a much desirable feature in a real-life quantitative investment and trading environment. In the circumstances, our results should go some way towards convincing a growing number of quantitative fund managers to experiment beyond the bounds of traditional statistical and neural network models. In particular, the strategies consisting of modelling in a first stage the linear component of a financial time series and then applying a neural network to learn its nonlinear elements and the use of Genetic Programming appear quite promising.

APPENDIX

A.1 ARMA Model

The output of the ARMA model used in this paper is presented below.

```
Dependent Variable: RETURNS
Method: Least Squares
Date: 03/17/09   Time: 22:18
Sample (adjusted): 8 1738
Included observations: 1731 after adjustments
Convergence achieved after 37 iterations
Backcast: 1 7
============================================================================
        Variable         Coefficient    Std. Error    t-Statistic     Prob.
============================================================================
           C               0.000290      0.000303      0.956602     0.3389
         AR(1)             0.375505      0.052705      7.124626     0.0000
         AR(3)            -0.244662      0.024991     -9.789999     0.0000
         AR(7)            -0.678906      0.044902     -15.11958     0.0000
         MA(1)            -0.374290      0.053055     -7.054702     0.0000
         MA(3)             0.269470      0.026409      10.20353     0.0000
         MA(7)             0.677169      0.044295      15.28785     0.0000
============================================================================
R-squared                  0.026582   Mean dependent var           0.000288
Adjusted R-squared         0.023194   S.D. dependent var           0.012549
S.E. of regression         0.012403   Akaike info criterion       -5.937710
Sum squared resid          0.265213   Schwarz criterion           -5.915645
Log likelihood             5146.088   F-statistic                  7.846483
Durbin-Watson stat         1.856760   Prob(F-statistic)            0.000000
============================================================================
Inverted AR Roots          .89-.44i      .89+.44i     .31-.92i     .31+.92i
                          -.54+.70i     -.54-.70i       -.93
Inverted MA Roots          .88-.45i      .88+.45i     .31-.92i     .31+.92i
                          -.54+.70i     -.54-.70i       -.94
============================================================================
```

A.2 Performance Measures

The performance measures are calculated as follows:

Performance Measure	Description	
Annualised Return	$R^A = 252 * \frac{1}{N} \sum_{t=1}^{N} R_t$ *with* R_t *being the daily return*	[14]
Cumulative Return	$R^C = \sum_{t=1}^{N} R_t$	[15]
Annualised Volatility	$\sigma^A = \sqrt{252} * \sqrt{\frac{1}{N-1} * \sum_{t=1}^{N} (R_t - \bar{R})^2}$	[16]
Information Ratio	$IR = \frac{R^A}{\sigma^A}$	[17]
Maximum Drawdown	*Maximum negative value of* $\sum (R_t)$ *over the period* $MD = \underset{i=1,\cdots,t;t=1,\cdots,N}{Min} \left(\sum_{j=i}^{t} R_j \right)$	[18]

Table 9: Trading simulation performance measures

A.3 Empirical Results in the Training and Test Sub-Periods

		NAIVE	MACD	ARMA	MLP	RNN	HONN
Information Ratio	***(excluding costs)***	1.55	1.24	1.24	1.57	1.53	1.61
Annualised Volatility	**(excluding costs)**	19.32%	19.49%	19.83%	19.60%	19.60%	19.59%
Annualised Return	***(excluding costs)***	29.86%	24.29%	24.66%	30.72%	30.02%	31.56%
Maximum Drawdown	***(excluding costs)***	-23.39%	-25.42%	-26.70%	-27.52%	-34.66%	-39.70%
Positions Taken	***(annualised)***	114	34	50	86	81	108

		Hybrid-MLP	Hybrid-RNN	Hybrid-HONN
Information Ratio	***(excluding costs)***	2.13	2.01	2.26
Annualised Volatility	**(excluding costs)**	19.42%	19.44%	19.40%
Annualised Return	***(excluding costs)***	41.35%	39.01%	43.77%
Maximum Drawdown	***(excluding costs)***	-37.20%	-26.86%	-37.20%
Positions Taken	***(annualised)***	102	79	77

	Mixed-MLP	Mixed-RNN	Mixed-HONN
Information Ratio ***(excluding costs)***	2.07	1.93	2.11
Annualised Volatility **(excluding costs)**	19.45%	19.47%	19.44%
Annualised Return ***(excluding costs)***	40.17%	37.57%	41.12%
Maximum Drawdown ***(excluding costs)***	-37.89%	-41.47%	-37.52
Positions Taken ***(annualised)***	46	68	47

	GP
Information Ratio ***(excluding costs)***	2.19
Annualised Volatility **(excluding costs)**	19.33%
Annualised Return ***(excluding costs)***	42.24%
Maximum Drawdown ***(excluding costs)***	-31.23%
Positions Taken ***(annualised)***	50

Table 10: In-sample trading performance

A.4 Genetic Programming Characteristics

We present below the characteristics of the Genetic Programming Algorithm with the best trading performance on the test sub-period.

Population Size:	200
Max tree depth:	6
Function Set:	+, -, *, /, ^, ^2, ^3, ^1/2, ^1/3, Exp, If,sin, cos, tan
Fitness evaluation function:	Mean Squared Error
Tournament Size:	4
Crossover trials:	1
Mutation Probability:	0,75

Table 11: Genetic Programming characteristics

A.5 Networks Characteristics

We present below the characteristics of the networks with the best trading performance on the test sub-period for the different architectures.

Parameters	MLP	RNN	HONN
Learning algorithm	***Gradient descent***	***Gradient descent***	***Gradient descent***
Learning rate	0.001	0.001	0.001
Momentum	0.003	0.003	0.003
Iteration steps	1500	1500	1000
Initialisation of weights	N(0,1)	N(0,1)	N(0,1)
Input nodes	11	11	11
Hidden nodes (1layer)	7	6	0
Output node	1	1	1

Table 12: Network Characteristics for Traditional Neural Networks

Parameters	Hybrid-MLP	Hybrid-RNN	Hybrid-HONNs

Learning algorithm	***Gradient descent***	***Gradient descent***	***Gradient descent***
Learning rate	*0.001*	*0.001*	*0.001*
Momentum	*0.003*	*0.003*	*0.003*
Iteration steps	*1500*	*1500*	*1000*
Initialisation of weights	*N(0,1)*	*N(0,1)*	*N(0,1)*
Input nodes	*13*	*13*	*13*
Hidden nodes (1layer)	*6*	*7*	*0*
Output node	*1*	*1*	*1*

Table 13: Network characteristics for Hybrid Neural Networks

Parameters	**Mixed-MLP**	**Mixed-RNN**	**Mixed-HONN**
Learning algorithm	***Gradient descent***	***Gradient descent***	***Gradient descent***
Learning rate	*0.001*	*0.001*	*0.001*
Momentum	*0.003*	*0.003*	*0.003*
Iteration steps	*1500*	*1500*	*1000*
Initialisation of weights	*N(0,1)*	*N(0,1)*	*N(0,1)*
Input nodes	*12*	*12*	*12*
Hidden nodes (1layer)	*6*	*7*	*0*
Output node	*1*	*1*	*1*

Table 14: Network characteristics for Mixed Neural Networks

REFERENCES

Adam, O., Zarader, L. and Milgram, M., (1994) ‘Identification and Prediction of Non-Linear Models with Recurrent Neural Networks’, *Laboratoire de Robotique de Paris.*

Andreou, P, C., Charalambous, C. and Martzoukos, H, S. (2006) ‘Knowledge Artificial Neural Networks to Enhanced Parametric Option Pricing’, *Research Paper Department of Public and Business Administration, University of Cyprus.*

Barricelli, N. A., (1954) ‘*Esempi numerici di processi di evoluzione’, Methodos*, 45-68.

Bishop, C., (1994) ‘Mixture Density Networks’. *Neural Computing Research Group Report*: NCRG/94/004, 1–25.

Box, G., Jenkins, G. and Gregory, G. (1994) ‘*Time Series Analysis: Forecasting and Control’, Prentice-Hall,* Hoboken**,** New Jersey.

Brown, G., Wyatt, J., Harris, R., and Yao, X. (2005), ‘Diversity Creation Methods: *A Survey and Categorization’*, Information Fusion, 6, 5–20.

Clemen, R. (1989), ‘Combining Forecasts: A Review and Annotated Bibliography’*, International Journal of Forecasting*, 5, 559–583.

Connor, J. and Atlas, L. (1993), ‘Recurrent Neural Networks and Time Series Prediction’, *Proceedings of the International Joint Conference on Neural Networks*, 301-306.

Cramer, N. L., (1985) ‘A Representation for the Adaptive Generation of Simple Sequential Programs’, in *Proceedings of an International Conference on Genetic Algorithms and the Applications,* Grefenstette, John J., (ed.), Carnegie Mellon University.

Dunis, C. and Huang, X. (2002), ‘Forecasting and Trading Currency Volatility: An Application of Recurrent Neural Regression and Model Combination’, *Journal of Forecasting*, 21, 5, 317-354.

Dunis, C., Laws, J. and Evans B. (2006a), ‘Trading Futures Spreads: An Application of Correlation and Threshold Filters’*, Applied Financial Economics*, 16, 1-12.

Dunis, C., Laws, J. and Evans B. (2006b), 'Modelling and Trading the Gasoline Crack Spread: *A Non-Linear Story', Derivatives Use, Trading and Regulation*, 12, 126-145.

Dunis, C., Laws, J. and Karathanasopoulos A. (2010a), 'Modelling and Trading the Greek Stock Market with Hybrid ARMA-Neural Network Models', CIBEF Working Papers. Available at www.cibef.com.

Dunis, C., Laws, J. and Karathanasopoulos A. (2010b), 'Modelling and Trading the Greek Stock Market with Mixed-Neural Network Models', CIBEF Working Papers. Available at www.cibef.com.

Dunis, C., Laws, J. and Sermpinis, G. (2008), 'Modelling and Trading the EUR/USD Exchange Rate at the ECB Fixing', *The European Journal of Finance,* 16, 6, 541 - 560

Dunis, C. and Huang, X. (2002), 'Alternative Volatility Models for Risk Management and Trading: Application to the EUR/USD and USD/JPY Rates', *Derivative Use, Trading & Regulation,* 11, 2, 126-156

Elman, J. L. (1990), 'Finding Structure in Time', *Cognitive Science*, 14, 179-211.

Fatima, S. and Hussain, G., (2008) 'Statistical Models of KSE100 Index Using Hybrid Financial Systems', *Neurocomputing*, 7, 2742-2746.

Fogel L. J., Owens A. J., and Walsh, M. J. (1964) 'On the Evolution of Artificial Intelligence', *Proceedings of the Fifth National Symposium on Human Factors in Electronics*, IEEE, San Diego, 63-76.

Fukunaga, A. and Stechert, A. (1998) 'Evolving Nonlinear Predictive Models for Lossless Image Compression with Genetic Programming', *Genetic Programming 1998: Proceedings of the Third Annual Conference*, Morgan Kaufmann, Wisconsin USA, 95 -102.

Fulcher, J., Zhang, M. and Xu, S., (2006) 'The Application of Higher-Order Neural Networks to Financial Time Series', *Artificial Neural Networks in Finance and Manufacturing, Hershey,* PA: Idea Group, London.

Ghiassi, M., Saidane, H. and Zimbra D. K. (2005), 'A Dynamic Artificial Neural Network Model for Forecasting Series Events', *International Journal of Forecasting*, 21, 341-362.

Giles, L. and Maxwell, T. (1987) ‘Learning, Invariance and Generalization in Higher Order Neural Networks’, *Applied Optics*, 26, 4972-4978.

Greg, T. and Hu, S. (1999), ‘Forecasting GDP Growth Using Artificial Neural Network’, *Working Paper, Bank of Canada*, 99-3.

Hansen, J. and Nelson, R., (2003) ‘Time-Series Analysis with Neural Networks and ARIMA-Neural Network Hybrids’, *Journal of Experimental and Theoretical Artificial Intelligence,* 15 (3), 315–330.

Hibbert, H., Pedreira, C. and Souza, R., (2000) ‘Combining Neural Networks and ARIMA Models for Hourly Temperature Forecast’, *Proceedings of International Conference on Neural Networks* (IJCNN 2000), 414–419.

Hibon, M. and Evgeniou. T., (2005) ‘To Combine or not to Combine: Selecting among Forecasts and their Combinations’*, International Journal of Forecasting*, 22, 15-24.

Holland J. H., (1975) ‘Adaptation in Natural and Artificial Systems’*, Ann Arbor, The University of Michigan Press.*

Kaastra, I. and Boyd, M. (1996), ‘Designing a Neural Network for Forecasting Financial and Economic Time Series’*, Neurocomputing*, 10, 215-236.

Kamijo, K. and Tanigawa,T. (1990), ‘Stock Price Pattern Recognition: A Recurrent Neural Network Approach’, *In Proceedings of the International Joint Conference on Neural Networks,* 1215-1221.

Karayiannis, N. and Venetsanopoulos, A. (1994), ‘On the Training and Performance of High-Order Neural Networks’, *Mathematical Biosciences*, 129, 143-168.

Knowles, A., Hussein, A., Deredy, W., Lisboa, P. and Dunis, C. L. (2009), ‘Higher-Order Neural Networks with Bayesian Confidence Measure for Prediction of EUR/USD Exchange Rate’, in M. Zhang [ed.] *Artificial Higher Order Neural Networks for Economic and Business,* Information Science Reference, London, 48-59.

Kosmatopoulos, E., Polycarpou, M., Christodoulou, M. and Ioannou, P., (1995) ‘High-Order Neural Network Structures for Identification of Dynamical Systems’, *IEEE Transactions on Neural Networks*, 6, 422-431.

Koza, J. R., (1990) ‘Genetic Programming: A Paradigm for Genetically Breeding Populations of Computer Programs to Solve Problems’, *Stanford University Computer Science Department*, Stanford

Koza, J. R., (1992) *Genetic Programming: On the Programming of Computers by Means of Natural Selection*, MIT Press, New York

Koza, J. R., (1994) ‘Genetic Programming II: Automatic Discovery of Reusable Programs’, MIT Press, New York

Koza J. R., (1998) ‘Genetic Programming’, In Williams, J. G.and Kent, A., [eds.]. *Encyclopedia of Computer Science and Technology*. New York, NY: Marcel-Dekker. 39, (Supplement 24), 29–43,

Koza, J. R., Bennett, F. H., Andre, D. and Keane, M. A., (1999) *Genetic Programming III: Darwinian Invention and Problem Solving,* Morgan Kaufmann, San Fransisco

Koza, J. R., Keane, M. A., Streeter, M. J., Mydlowec, W., Yu, J. and Lanza, G. (2003) *Genetic Programming IV: Routine Human-Competitive Machine Intelligence*, Kluwer Academic Publishers, New York

Lisboa, P. J. G. and Vellido, A. (2000), ‘Business Applications of Neural Networks’, vii-xxii, in P. J. G. Lisboa, B. Edisbury and A. Vellido [eds.] *Business Applications of Neural Networks: The State-of-the-Art of Real-World Applications,* World Scientific, Singapore,.

Madár, J., Abonyi, F. and Szeifert, F. (2005) ‘Genetic Programming for the Identification of Nonlinear Input-Output Models’, Industrial and Engineering Chemistry Research. p.o Box 158 Veszprem 8201 Hungary

Madár, J., Abonyi, F. and Szeifert, F. (2004) ‘Genetic Programming for System Identification’*, Intelligent Systems Design and Applications* (ISDA). University of Veszprem, Hungary

Makridakis, S., Anderson A., Carbone, R., Fildes, R., Hibdon, M., Lewandowski, R., Newton, J., Parzen, E. and Winkler, R. (1982) ‘The Accuracy of Extrapolation (Time Series) Methods: Results of a Forecasting Competition’, *Journal of Forecasting*, 1, 111–153.

Makridakis, S., (1989) ‘Why Combining Works?’, *International Journal of Forecasting*, 5, 601–603.

Newbold, P. and Granger, C. W. J. (1974) ‘Experience with Forecasting Univariate Time Series and the Combination of Forecasts (with discussion)’, *Journal of Statistics*, 137, 131–164.

Palm, F. C. and Zellner, A., (1992), ‘To Combine or not to Combine? Issues of Combining Forecasts’, *Journal of Forecasting,* 11, 687–701.

Pindyck, R. and Rubinfeld, D. (1998), *Econometric Models and Economic Forecasts,* 4th edition, McGraw-Hill, New York.

Psaltis, D., Park, C. and Hong, J. (1988), ‘Higher Order Associative Memories and their Optical Implementations’*, Neural Networks,* 1, 149-163.

Rechenberg, I. (1971) ‘Evolutionsstrategie - Optimierung technischer Systeme nach Prinzipien der biologischen Evolution’, (*PhD thesis*). Reprinted by Fromman-Holzboog (1973). ISBN 3-7728-03-73-3

Redding, N., Kowalczyk, A. and Downs, T. (1993), ‘Constructive Higher-Order Network Algorithm that is Polynomial Time’*, Neural Networ*ks, 6, 997-1010.

Shapiro, A. F. (2000), ‘A Hitchhiker’s Guide to the Techniques of Adaptive Nonlinear Models’, *Insurance, Mathematics and Economics,* 26, 119-132.

Tenti, P. (1996), ‘Forecasting Foreign Exchange Rates Using Recurrent Neural Networks’, *Applied Artificial Intelligence*, 10, 567-581.

Terui, N. and van Dijk, H. (2002), ‘Combined Forecasts from Linear and Nonlinear Time Series Models’, *International Journal of Forecasting*, 18, 421–438.

Theil, H. (1996), ‘Applied Economic Forecasting, North-Holland, Amsterdam, Netherlands.

Tino, P., Schittenkopf, C. and Doffner, G. (2001) ' Financial Volatility Trading Using Recurrent Networks' , IEEE transactions in Neural Networks, 12, 4, 856-874.

Tsang, E. P. K., Butler J. M. and Li, J. (1998) ‘EDDIE Beats the Bookies’, *Journal of Software* – Practice and Experience, Wiley, 28, (10). 1033-1043

Tseng, F. M., Yu, H. C. and Tzeng, G. H. (2002) ‘Combining Neural Network Model with Seasonal Time Series ARIMA Model’, *Technological Forecasting and Social Change*, 69, 71–87.

Wang, Y. F., (2007) ‘Nonlinear Neural Network Forecasting Model for Stock Index Option Price: Hybrid GJR-GARCH Approach’, *Expert Systems with Applications*, 36, 564-570.

Werner, J. C. and Fogarty, T. C. (2001) ‘Genetic Programming Applied to Collagen Disease & Thrombosis’, *South Bank University*, London.

Willis, M. J., Hiden, H. G., Marenbach, P., McKay, B, and Montague, G. A. (1997)’ Genetic Programming: An Introduction and Survey of Applications’, *Second International Conference on Genetic Algorithms in Engineering Systems*, 65,314 – 319.

Winkler, S., (2004) ‘Identifying Nonlinear Model Structures Using Genetic Programming’, Diploma Thesis, Institute of Systems Theory and Simulation, Johannes Kepler University, Linz, Austria.

Winkler, S., Affenzeller, M. and Wagner, S. (2004a) ‘New Methods for the Identification of Nonlinear Model Structures Based Upon Genetic Programming Techniques’. *Proceedings of the 15th International Conference on Systems Science*, 1, 386-393.

Winkler, S., Affenzeller, M. and Wagner, S. (2004b) ‘Identifying Nonlinear Model Structures Using Genetic Programming Techniques’, *Cybernetics and Systems* 2004, 1 689-694.

Zhang, M., Xu, S., X. and Fulcher, J. (2002), ‘Neuron-Adaptive Higher Order Neural-Network Models for Automated Financial Data Modelling’, *IEEE Transactions on Neural Networks*, 13, 1, 188-204.

Zhang, G. P., (2003) ‘Time Series Forecasting Using a Hybrid ARIMA and Neural Network Model’, *Neurocomputing,* 50, 159–175.

Zhang, G. P., and Qi, M., (2005) ‘Neural Network Forecasting for Seasonal and Trend Time Series’*, European Journal of Operational Research*, 160 (2), 501–514.