Modelling and Trading the Greek Stock Market with Gene Expression and Genetic Programing Algorithms

by

Andreas Karatahansopoulos*

Georgios Sermpinis**

Jason Laws***

Christian Dunis****

October 2013

Abstract

This paper presents an application of the gene expression programming (GEP) and integrated genetic programming (GP) algorithms to the modelling of ASE 20 Greek index. GEP and GP are robust evolutionary algorithms that evolve computer programs in forms of mathematical expressions, decision trees or logical expressions. The results indicate that GEP and GP produce significant trading performance when applied to the ASE 20 and outperform the well-known existing methods. The trading performance of the derived models is further enhanced by applying a leverage filter.

Keywords

Gene Expression Programming Algorithm, Leverage, Quantitative Trading Strategies, Genetic Programming, Evolutionary Algorithms.

Andreas Karathanasopoulos Royal Docklands Business School, University of East London, (E-mail:a.karathanasopoulos@uel.ac.uk)

Georgios Sermpinis University of Glasgow Business School, University of Glasgow, Adam Smith Business School (Email: georgios.sermpinis@glasgow.ac.uk)

Christian Dunis Horus Partners Wealth Management Group SA, Genève, Suisse and Liverpool Business School, JMU, John Foster Building , Liverpool L3 5UZ (E-mail: christian.dunis@hpwmg.com)

Jason Laws University of Liverpool Management School, The University of Liverpool, Chatham Street, Liverpool, L69 7ZH (E-mail: J.Laws@liverpool.ac.uk)

1. INTRODUCTION

Developing highly accurate techniques for predicting financial time series is a crucial problem for economists and financial researchers. The traditional statistical methods seem to fail to capture the discontinuities, the nonlinearities and the high complexity of datasets such as financial time series. Evolutionary algorithms have the desirable characteristic of learning capacity and are hence more likely to capture the complex non-linear models which are dominant in the financial markets (see Chen (2002)). However their limitations, such as the bloat effect, and contradictory empirical evidence surrounding their pattern recognition capabilities often create scepticism about their use among practitioners.

Genetic programming (GP) and gene expression (GEP) algorithms are domain-independent problem-solving techniques that are run in various environments. GP and GEP can be categorized in the forecasting bracket known in the finance world as 'Evolutionary Algorithms'. These environments are structured in a manner which approximates problems in order to produce forecasts at a high degree of accuracy. The basis for this type of problem solving techniques derives from the Darwinian principle of reproduction and survival of the fittest. Additionally, they can be considered similar to the biological genetic operations such as crossover recombination and mutation. These algorithms have been successfully applied to many real world problems (Divsalar et al. 2011, 2012; Yang et al. 2012; Gandomi et al. 2013; Gandomi and Alavi 2011; Alavi and Gandomi 2011; Zargari et al. 2012; Gandomi et al. 2011; Alavi et al. 2011; Yaghouby et al. 2010). It can be argued that the GEP proves to be superior to GP due to the mere fact that it clearly distinguishes the differences between the genotype[1] and the phenotype[2] of individuals within a population. For instance, whilst a traditional Genetic Algorithm (GA) classifies individuals as symbolic strings of fixed size (i.e. chromosomes) and GP classifies its individuals as non-linear comprising of different shapes and sizes (tree like structures); the GEP encompasses a combination of both. Hence, Ferreira (2001) stresses that GEP represents not only an individual's genotype, in the form of chromosomes, but also its phenotype as a tree like structure of expressions in order to establish fitness. Compared to Neural Networks (NNs), GEP eradicate the risk of getting trapped in a local optima and seem able to reach the optimal solution faster.

The motivation for this paper is to investigate the financial forecasting performance of a GEP and an integrated GP environment, using only autoregressive terms as inputs. This is achieved by benchmarking their trading performance, with an application to forecasting the one day ahead return of the Greek stock market index, with a Multi-Layer Perceptron (MLP) of NNs, an autoregressive moving average (ARMA), a moving average model (MACD) and a naïve trading strategy. Furthermore, the paper explores whether the application of a simple leverage strategy can improve the trading performance of the proposed models. The variable being forecast in this paper is the one day ahead forecast for the Athens Stock Exchange (ASE) market return. The ability to successfully generate such forecasts suggests that it should be possible to implement a simple trading strategies where a trader would buy (or sell) the market depending on whether a positive (or negative) forecast is generated. The development of derivative securities, such as stock index futures, greatly facilitates such strategy and has reduced transaction costs. Leverage is then applied to exploit the high information ratios of our model and further improve the trading performance of our models. Our findings support this view as the results show that in a simple trading simulation exercise on the Athens stock exchange, the GEP algorithm outperforms all other models.

The remainder of the paper is organised as follows. Section 2 presents some of the existing literature relevant to stock market prediction. Section 3 provides an overview of the GEP and GP algorithms. Section 4 describes the

dataset used for this research and its characteristics while Section 5 discusses the details and the model development for GEP and GP algorithms along with our benchmarks. Section 6 displays the empirical results of all the models considered and investigates the possibility of improving their performance with the application of leverage. Section 7 provides some concluding remarks.

2. LITERATURE REVIEW

Stock market prediction has been the focus of researchers, decision makers and economists for almost a century. In his seminal paper Cowles (1933) constructs a portfolio using Dow Jones industrial average (DJIA) market index forecasts and assess its performance. Since then practitioners and researchers have produced hundreds of mathematical models in an effort to replicate the stock markets patterns. The focus of this research over the last years has moved towards artificial intelligence models, such as GP and Neural Networks (NNs). The motivation of this paper is to apply a GEP algorithm to the task of forecasting and trading the ASE 20 stock market index. Although there have been several applications of NNs and GP in financial forecasting, the empirical evidence around GEP is limited. This can partly be explained by the complexity of the GEP algorithm compared to other artificial intelligence models (see Ferreira (2001)). Nevertheless, GEP has provided promising empirical evidence in other fields of science such as computing and mining. See, for example, Lopez and Weinert (2004), Margny and El-Semman (2005) and Dehuri and Cho (2008).

Donaldson and Kamstra (1996) used Neural Network's to combine volatility forecasts of the US, Canadian, Japanese and UK stock markets. Their results demonstrate the superiority of Neural Network's over traditional linear models More recently, Kaboudan (2000) applied successfully GP in predicting the daily highest and lowest stock prices of six US stocks (Citigroup, Compaq Computers, General Electric, Pepsi, Sears and Microsoft). They find that the return from investment (ROI) from trading decisions based on GP forecasting is greater than ROI from investment decisions based on NNs forecasting in five of the six stocks, though they do not consider risk adjusted returns, such as an information ratio, as this paper does. In addition Nag and Mitra (2002) combined Neural Networks with GP to forecast several exchange rates with impressive results. Likewise Kwon and Moon (2003) combined NNs and GP to trade successfully 36 stocks over a 9 year period. Grosan *et. al.* (2005) and Grosan and Abraham (2006) compared a Genetic multi-expression programming algorithm with NNs and Support Vector Machine models in two forecasting exercises over several stock indices. The results of both studies strongly supported the use of a genetic model. Alvarez-Diaz and Alvarez (2005) forecast the GBP/USD and JPY/USD with a GP and NNs models with, unfortunately, ambiguous results. Almanza and Tsang (2007) used GP to accurately detect the stock price movements and Chiu and Chen (2009) combined Support Vector Machines with GP to estimate Taiwanese stock and futures prices. More recently, Lee and Tong (2011) combined GP with NNs and an ARMA model to accurately forecast the US GDP while Sermpinis *et.*

al. (2012) traded successfully the EUR/USD exchange with several different artificial intelligence models.

Other recent literature in the area includes Matias and Reboredo (2012) who forecast the S&P 500 intraday stock market returns with a range of linear (simple autoregressive and random walk models) and non linear models (such as smooth transition, Markov switching, NNs, nonparametric kernel regression and support vector machine) models. Their results indicate the superiority of the non linear models in both statistical and economic terms. In addition Bekiros and Georgoutsos (2008) utilise Recurrent NNs to forecast and trade successfully the directional change of NASDAQ index with RNN.

3. Genetic Programming and Genetic Expression Framework

3.1 The Genetic Programming Framework

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 the primary focus is 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 with the GP application there is a significant 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. The 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 GA.

In our application of the genetic programming formulas are utilized 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) in figure 1, is characterized by the algebraic expression $4.0/x_1(t\text{-}1) + \ln(x_2(t\text{-}2))$. In this case there is one output and the terminal nodes are constant at 4. Additionally, the outputs are expressed by $x_1(t\text{-}1)$ and $x_2(t\text{-}2)$. 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.

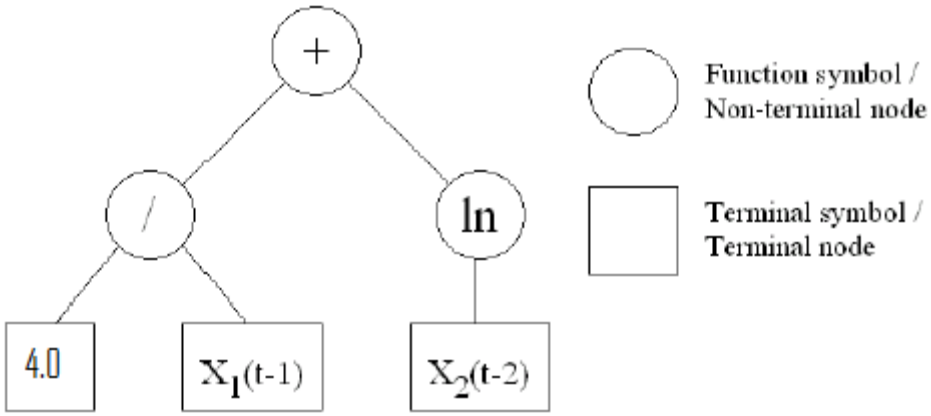


Fig. 1: Example of a tree structure

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. In the majority of these models produced in the initial population are 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 mean squared 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 the 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 based on minimising the mean squared error (MSE).

(5) Having identified the tournament winners in the previous step the algorithm then proceeds 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 generated. 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. A mutation tree structure example is presented in figure 5 below.

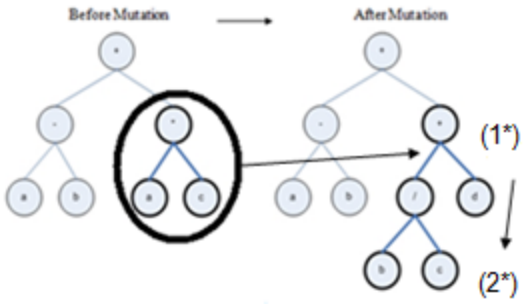


Fig. 2: Mutation tree structure example

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. The reader is directed to Koza (1992) for more details of this procedure. Figure 6 presents a crossover family tree like structure example.

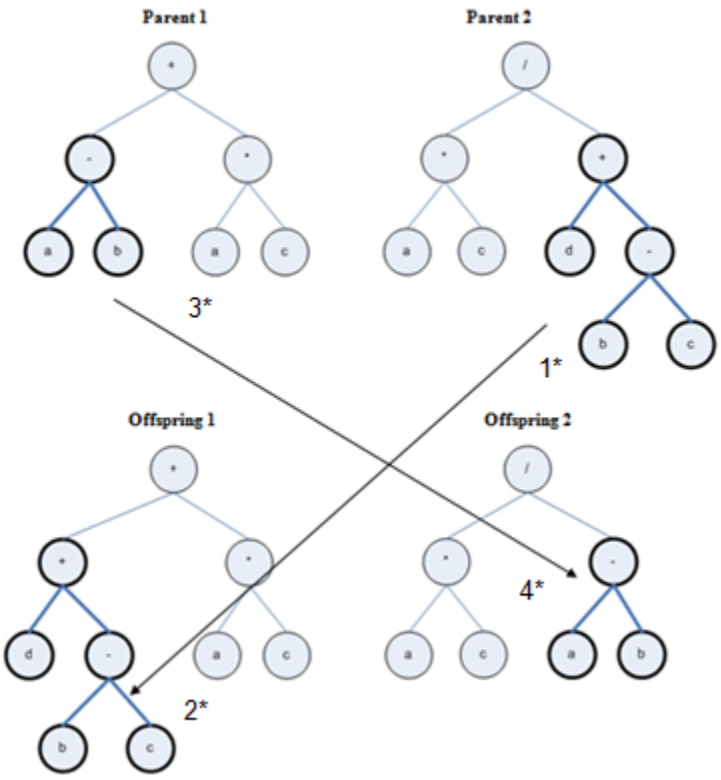
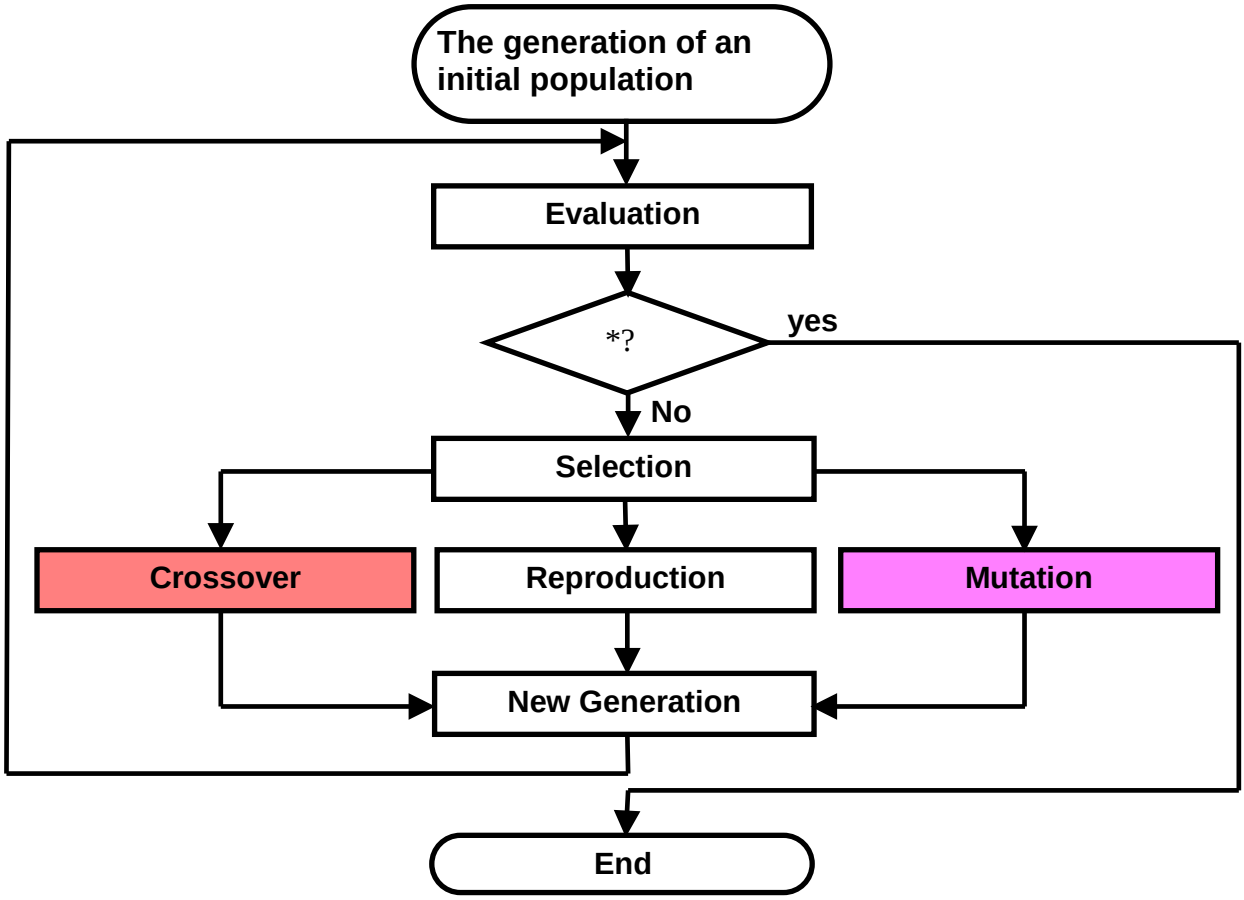


Fig. 3: Crossover family tree like structure example

(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 this paper the termination criterion is set to 100,000 at which point the cycles are stopped and forecasted results can be obtained.
(8) This procedure ultimately produces the best individual (model) of the population as a result.

A summary of the methodology described above is depicted in figure 7 below.



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

Fig. 4: The architecture of Genetic Programming Algorithm

3.2 The Gene Expression Programming Framework

As mentioned before the models in GEP are symbolic strings of fixed length representing an organism's genome (chromosome/genotype), but these simple entities are encoded as non linear entities of different sizes and shapes determining an organism's fitness (expression trees/phenotype). GEP chromosomes are made up of multiple genes spanning equal lengths across the structure of the chromosome. Each gene is comprised of a head (detailing symbols specific to functions and terminals**)** and a tail (which only includes terminals). For a mathematical representation please refer to equation [1] below:

$$t = (n-1)h+1 \qquad [1]$$

Where:
h = the head length of the gene.
t = the tail length of the gene.

n = total number of arguments within the function[3] (maximum arity)

The set of terminals included within both the heads and tails of the chromosomes contain constants as well as case specific variables. In addition, regardless of the fact that each of the genes is equal and fixed in size they hold the same capacity to code for multiple and varied expression trees (ET). For example, the structure of GEP is able to cope in circumstances when the first element of a gene is terminal producing a single node as well as when multiple nodes ('sub-trees' reproduced by functions) are produced in search for eventual terminality. In contrast with its predecessors, GEP does not require the rejection of invalid individuals from the population, as valid ETs are always generated. Thus, each gene encodes an ET and in situations where multiple generation arise, GEP codes for sub ETs with interlinking functions to enable reproduction of generations**.** Furthermore, the expression of each ET is enabled by an Open Reading Frame (ORF) which assists in the decoding process. Additionally, while the ORF is initiated at the beginning of each gene it has to be understood that the eventual terminal points are not always determined to be located at the end of the gene.

Although it is crucial to understand the workings of a GEP it is also just as important to understand the step by step process of evolution. This is depicted in figure 8 below.

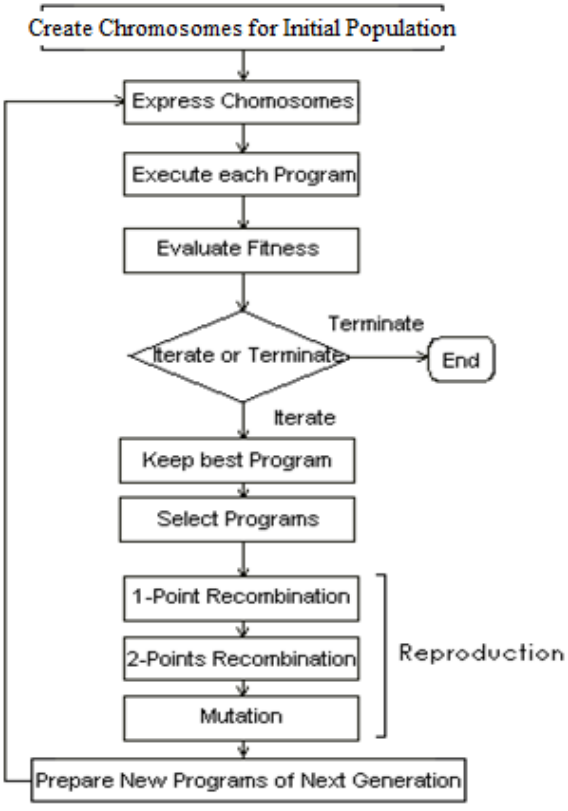


Fig. 5: Flow chart of Gene Expression Algorithm

[3] This is determined by the user. In most cases a function will either be a Boolean function or another mathematical function that is suited to a specific problem.

The different steps of the algorithm from the above diagram are explained in more detail as follows:

1. Creation of Initial Population
Similar to other evolutionary algorithms, GEP randomly generates an initial population from populations of individuals and all succeeding populations are spawned from this initial population. In the spawning of new generations genetic operators evolve each of the individuals by 'mating' them with other individuals in the population. These genetic operators are deciphered by the nature of the problem which one wants to solve. Genetic operators may include (but are not limited to) '+', '-', '*' and '/' symbols for mathematical models and 'And', 'Or', 'Nand', 'Nor', 'Xor', 'Nxor', '<', '>', '< or =', and '> or =' for logical expressions as explained by Ferreira (2001). Therefore, the terminals and functions (symbols) may vary from problem to problem. Other characteristics such as gene size also have to be specified by the user at this stage.

2. Express chromosomes
In this step the model progresses by developing expression trees from our chromosomes. The structure of each ET is in such a way that the root or the first node corresponds with beginning of each gene. The resulting offspring evolved from the first node is dependent on the number of arguments. In this process of evolution the functions may have numerous arguments however the terminals have a parity of zero. Each of the resulting offspring's characteristics is populated in nodes ordered from left to right. This process is concluded once terminal nodes are established.

3. Execute each program
The next step of the algorithm is to generate the initial population and develop resulting ETs. This process is explained further in Ferreira (2001).

4. Evaluate fitness
In order to create an accurate model suited to our forecasting requirements it is imperative that a function which minimizes error and improves accuracy is used. In our application, as with the GP algorithm, the fitness value is defined as the MSE with the lowest MSE being targeted as the best.

5. Keep best Program
In our GEP model the main principal during the process of evolution is the generation of offspring from two superior individuals to achieve 'elitism'. As a consequence the best individuals from the parent generation produce offsprings in future generations with the most desirable traits whilst the individuals with less desirable traits are removed. On this basis our model minimizes error and maintains superior forecasting abilities. As explained in greater detail by Ferreira (2001), elitism is the cloning of the best chromosome(s)/individual(s) to the next population (also called generation). Furthermore, the role of 'elitism' enables the selection of fitter individuals without eliminating the entire population.

6. Selection

The selection of individuals based on their 'fitness' is carried out during the 'tournament' selection for reproduction and modification. This process selects the individuals at random with the superior ones being chosen for genetic modification in order to create new generations. The intensity of competition is dictated by the tournament size which is adjusted and set by the practitioner. The greater the tournament size then the more competitive the selection process and therefore weaker individuals are less likely to compete.

7. Reproduction
In the reproduction of future generations to the algorithm considers the types of genetic operators which make this 'evolution' possible. Specifically, genetic operators, known as mutation and recombination, are applied. These are explained below.

Mutation: This is the creation of a new model that is mutated randomly from an existing one. Firstly a parent is randomly selected with a probability related to its fitness. Then the mutation point on the parent's chromosome is indiscriminately chosen as an independent point. Afterwards the mutation randomly changes one or more genes representing part of the solution it encodes and the new mutated individual is added to the population. This procedure is described in detail in Ferreira (2006).

Recombination: in contrast to our mutation operator this process is not executed at random. Instead the parent chromosomes are matched and split up or 'spliced' at identical points in order to determine recombination points. The subsequent spliced parts of each of the genes are then exchanged between the two selected chromosomes on the basis of probability. This results in two new individuals as a result of genetic engineering. Note that during reproduction it is the chromosomes of the individuals, not the expression trees that are reproduced with modification and passed onto the next generation.

8. Prepare new programs of the next generation
At this step, the algorithm replaces the tournament losers with the new individuals created by reproduction in the population.

9. Termination criterion
At this point a check is made to determine whether the termination criterion is fulfilled, if it is not we return to step 2. As a termination criterion we used a maximum number of 100,000 generations during which the GEP was left to run.

10. Results
As a result the model returns the best individual found during the evolution process.

4. THE ASE 20 GREEK INDEX AND RELATED FINANCIAL DATA

For futures on the ASE 20 that are traded in derivatives markets, the underlying asset is the blue chip index ASE-20. The 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 already an established benchmark. It represents over 50% of the ASE's total market capitalisation and as with many modern stock indexes has a heavier weighting 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. 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. Whilst the futures contract is traded in index points, as with all stock index futures the monetary value of the contract is calculated by multiplying the futures price by the value of each index point which for the FTSE/ASE20 index is 5 EUR per point. For example, a contract trading at 1,400 points is valued at 7,000 EUR.

This paper examines 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. Table 1 presents illustrates how the dataset is divided:

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

Figure 2 depicts the ASE 20 series for the period under consideration and illustrates that this period has largely consisted of three significant trends: A downtrend from 01/01 to 04.03, followed by a long up-trend from 04/03 to 11/07, with a number of significant market corrections, followed by a downtrend from 11/07 to 12/08.

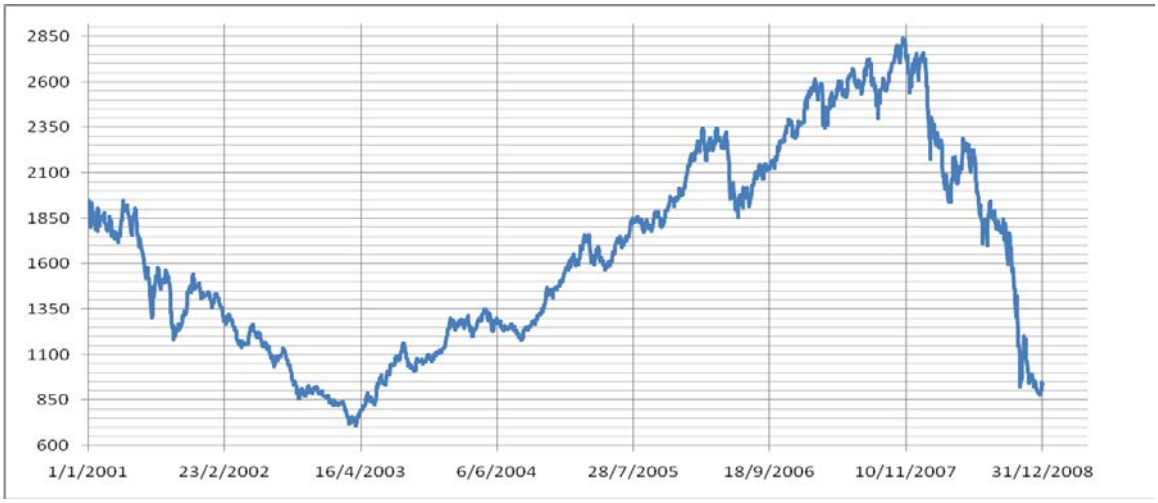


Fig. 6: ASE 20 fixing prices (total dataset).

The observed ASE 20 time series is non-normal (Jarque-Bera statistics confirms this at the 99% confidence level) containing slight skewness and high kurtosis. It is also non-stationary and hence the data was transformed into a stationary daily series of rates of arithmetic return[4].

The summary statistics of the ASE 20 returns series are presented in Figure 3 below. In this paper it is this return series that will be forecasted from our models.

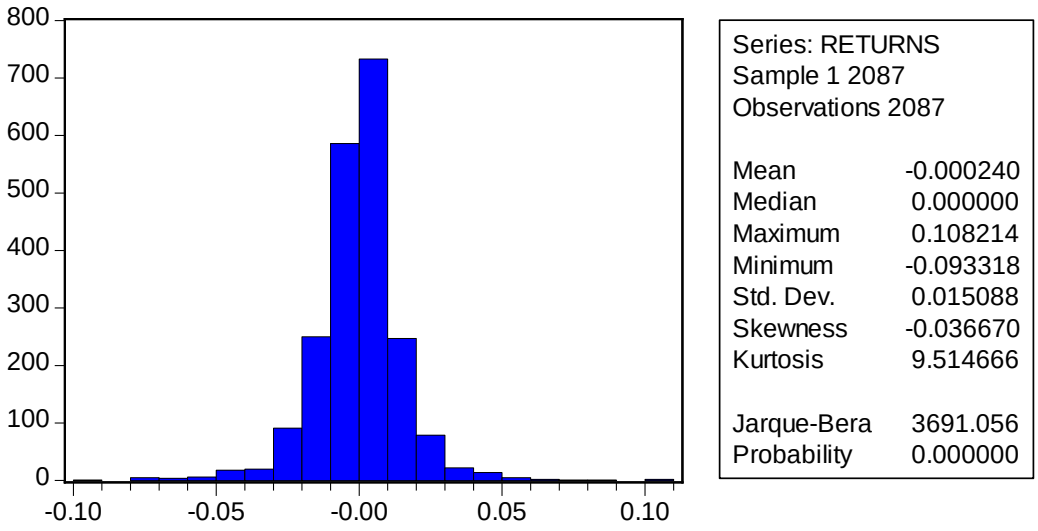


Fig. 7: ASE 20 returns summary statistics (total dataset).

The inputs to our GP, GEP and MLP algorithms are based on the autocorrelation function and some ARMA experiments. The outcome of this selection process is a set of autoregressive and moving average terms of the

[4] Confirmation of its stationary property is obtained at the 1% significance level by both the Augmented Dickey Fuller (ADF) and Phillips-Perron (PP) test statistics.

ASE 20 returns and the 1-day Riskmetrics volatility series[5]. This set is detailed in table 2 below.

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 GEP and GP

In order to train our MLP algorithm the dataset was further divided as detailed in Table 3 below:

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 3: The neural networks datasets

5. FORECASTING MODELS

In order to benchmark the trading performance of our GP and GEP models tree standard benchmarks (a naïve strategy, a moving average cross over and divergence model and an ARMA model) are utilised. In addition we utilised a standard MLP model. The performance of each strategy is evaluated in terms of trading performance via a simulated trading strategy. The section below presents the technical characteristics of our GP and GEP algorithms along with our benchmarks.

5.1 Naïve strategy

The naïve strategy simply takes the most recent period change as the best prediction of the future change.

5.2 MACD

In order to implement the MACD strategy two moving average series are created with different moving average lengths. The decision rule for taking positions in the market is as follows:

[5] The RiskMetrics volatility model was developed by JP Morgan in 1989 and is used extensively in the literature to benchmark trading performance.

- Positions are taken if the moving averages intersect. 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[6].

A number of differing moving average lengths combinations were considered and the best pairing 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.

5.3 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)[7].

Using as a guide the correlogram in the training and the test sub periods a restricted ARMA (7, 7) model was selected. 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 A.1).

The selected ARMA model takes the form:

$$Y_t = 2.90 \cdot 10^{-4} + 0.376 Y_{t-1} - 0.245 Y_{t-3} - 0.679 Y_{t-7} + 0.374 \mu_{t-1} - 0.270 \mu_{t-3} - 0.677 \mu_{t-7} \quad [2]$$

5.4 The MLP Model Architecture

NNs 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 nodes in this layer corresponds to the number of explanatory variables). The last layer is called the output layer (the number of nodes in this layer corresponds to the number of response variables). An intermediary layer of nodes, the hidden layer, separates the input from the output layer (the number of nodes in this layer 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

[6] 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.

[7] For a full discussion on the procedure, refer to Box et al. (1994).

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[8] (Shapiro (2000)). The learning algorithm simply tries to find those weights which minimize an error function (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 is usually achieved in the literature 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 a fitness function in the test dataset. Finally, the predictive value of the model is evaluated applying it to the validation dataset (out-of-sample dataset).

The parameters setting and architecture was optimized through a sensitivity analysis in the in sample period. The set of parameters that provided the higher trading performance in the test sub-period was then selected.

The network architecture of a 'standard' MLP looks as presented in figure 4[9]:

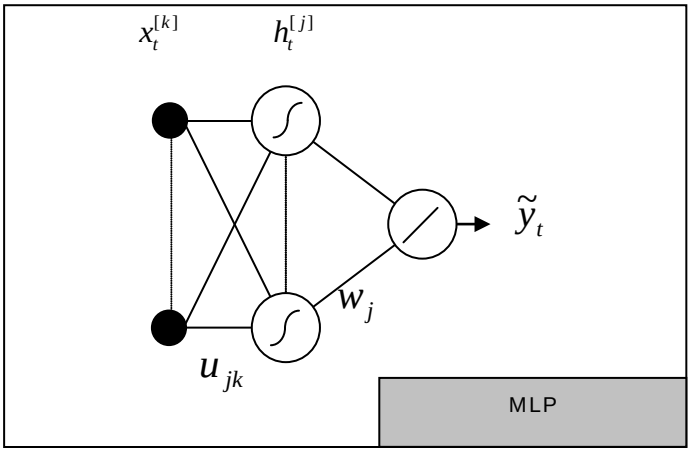


Fig. 8: A single output, fully connected MLP model

Where:

[8] 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)).
[9] The bias nodes are not shown here for the sake of simplicity.

$x_t^{[n]}\ (n = 1,2,\cdots,k+1)$ 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

u_{jk} and w_j are the network weights

$\int$ is the transfer sigmoid function: $S(x) = \frac{1}{1+e^{-x}}$, [3]

$/$ is a linear function: $F(x) = \sum_i x_i$ [4]

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 [5]

A summary of our MLP algorithm is presented in table 4 below.

Parameters	MLP
Learning algorithm	Gradient d
Learning rate	0.0
Momentum	0
Iteration steps	
Initialisation of weights	
Input nodes	
Hidden nodes (1 layer)	
Output node	

Table 4: MLP Characteri

The MLP was programmed using Java algorithm for modeling and trading a historical data, does not require mor Pavilion p6-2175ea Desktop PC.

5.5 Genetic Programming and G

The GP and the GEP codes we times with random seeds and t sample performance in t characteristics of our GP a development are provide

5.5.1 The GP Model A

The parameters used for the optimization of our individual models are presented in table 5 below.

Population Size:	200
Max tree depth:	6
Constants' range:	[-3, 3]
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 5: Genetic Programming Characteristics

We wrote the GP code in Java. We ran the model 40 times with random seeds and selected our best model based on their in sample performance.The time needed for each run was approximately 15 minutes using a standard HP Pavilion p6-2175ea Desktop PC. Equation [6] below presents our best model:

output=cos((x7) / ((-2.862)^(((1.0119) - (((((x11+x8)^(x1)) * (exp(((-2.8624) + (exp(((((-1.9846) / ((0.8175) * (1.2320))) + (1.2320)) - ((x2+x9) * ((-2.8624) + ((x4) * (((-2.3000) / ((-1.9846) * ((x5)^1/2))) + (tan(x3+x7)))))))^1/2))) * ((-2.3000)^2))))^3) / (tan(x9+x7)))) * (x6+x10)))) [6]

where x_i (i=1…11) are the model inputs and the output is our model forecast. It is recognised that the functions tan, cos and exp dominate our proposed forecasting equation. Unfortunately standard significance tests on equation [6] are not possible due to the non-linear nature of our GP algorithm (see Chen (2002)).

5.5.2 The GEP Model Architecture

A summary of the characteristics of our GEP algorithm is presented in table 6 below.

Population Size:	1000
Head length:	6
Constants' range:	[-3, 3]
Function Set:	+, -, *, /, ^, ^2, ^3, ^1/2, ^1/3, Exp, If,sin, cos, tan
Fitness evaluation function:	Mean Squared Error
Tournament Size:	20
Type of recombination:	Two point
Mutation Probability:	0,75

Table 6: Gene Expression Programming Characteristics

We wrote the GEP code in Java. We ran the model 40 times with random seeds and selected our best model based on their in sample performance.The average time required for each run was approximately 18 minutes using a standard HP Pavilion p6-2175ea Desktop PC. Equation [7] below presents our best model:

output=tan(((exp(((x1+x2)^1/2) + ((cos(0.9395))^3))) / (exp(((2.2580)^3) * (((((x6)^2) / (sin((x11) * (tan(exp((-0.6441)^((((x7) / (x4)) - (sin(((tan((-0.9604)^((cos(cos(x7))) / (x5+x8)))) + (1.9056)) – ((x3)^1/3))))^2)))))))^1/2)^2)))) * (x7)) [7]

where x_i (i=1…11) are the model inputs and the output is our model forecast. Again application of standard significance tests on equation [7] is prohibited by the non linear nature of our GEP algorithm.

6 EMPIRICAL TRADING SIMULATION RESULTS

6.1 Trading Performance

Our trading strategy applied is simple and identical for all the models. That is, go or stay long when the forecast return is above zero and go or stay short when the forecast return is below zero. Transaction costs of 0.14% per position are assumed.[10].

The trading performance measures, and their calculation description, are presented in Appendix A.2. Table 7 below presents the trading performance of our models in the in-sample period.

[10] 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 the average deal size (1 million Euros) gives us an average transaction cost for large “players” of 14 basis points or 0.14% per position.

	NAIVE	MACD	ARMA	MLP	GP Algorithm	Gene Expression
Information Ratio (excluding costs)	1.55	1.24	1.24	0.60	2.19	2.34
Annualised Volatility (excluding costs)	19.32%	19.49%	19.83%	38.11%	19.33%	19.31%
Annualised Return (excluding costs)	29.86%	24.29%	24.66%	22.99%	42.24%	45.19%
Maximum Drawdown (excluding costs)	-23.39%	-25.42%	-26.70%	-36.26%	-31.23%	-28.07%
Positions Taken (annualised)	114	34	50	61	50	52
Transaction costs	15.96%	4.76%	7.00%	8.54%	7.00%	7.28%
Annualised Return (including costs)	3.36%	19.53%	17.66%	14.45%	35.24%	37.91%

Table 7: Trading performance results – In-Sample

From the table above, we note that all our models present a positive trading performance after transaction costs. Our Gene Expression algorithm produces the more profitable forecasts in the in-sample period with an annualised return of 37.91% after transaction costs. The GP presents the second best performance with a 35.24% annualised return. On the other hand, the performance of our third non linear model, the MLP, is disappointing as it produce lower annualised return than our linear MACD and ARMA strategies. In table 8 below we present the trading performance of our models in the out-of-sample period.

	NAIVE	MACD	ARMA	MLP	GP Algorithm	Gene Expression
Information Ratio (excluding costs)	0.32	0.46	0.20	0.60	1.03	1.16
Annualised Volatility (excluding costs)	36.70%	38.12%	38.13%	38.11%	38.04%	38.01%
Annualised Return (excluding costs)	11.42%	17.63%	7.68%	22.99%	39.33%	44.16%
Maximum Drawdown (excluding costs)	-49.41%	-50.63%	-36.50%	-36.26%	-29.45%	-28.20%
Positions Taken (annualised)	119	38	72	105	67	71
Transaction costs	15.47%	4.94%	9.36%	13.65%	9.40%	9.94%
Annualised Return (including costs)	-4.05%	12.69%	-1.68%	9.35%	29.93%	34.22%

Table 8: Trading performance results – Out-of-Sample

We can see that, after transaction costs, the GEP model continues to outperform all the other strategies based on the annualized return of 34.22%. It is closely followed by the GP Algorithm strategy with a 29.93% annualized return. What is more remarkable is the extent to which they outperform the traditional MLP which has an annualized return of 9.35% after transaction costs. On the other hand, the naïve strategy and the ARMA model produce

negative results after transaction costs are taken into account. It is also worth noting the impressive performance of our linear MACD strategy with the third more profitable forecasts in both in-sample and out-of-sample periods. This is not surprising given the trending nature of our dataset.

6.2 Leverage to exploit high Information 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 and the ARMA models, 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%[11] 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[12]). Our final results are presented in table 9 below.

	NAIVE	MACD	ARMA	MLP	GP Algorithm	Gene Expression
Information Ratio (excluding costs)	0.32	0.70	0.20	0.60	1.03	1.16
Annualised Volatility (excluding costs)	36.70%	40.03%	38.13%	40.28%	41.84%	39.15%
Annualised Return (excluding costs)	11.42%	18.51%	7.68%	24.30%	43.26%	45.48%
Maximum Drawdown (excluding costs)	-49.41%	-53.16%	-36.50%	-38.32%	-31.02%	-29.04%
Leverage Factor	-	1.050	-	1.057	1.1	1.03
Positions Taken (annualised)	119	38	72	105	67	71
Transaction and leverage costs	15.47%	4.94%	9.36%	13.65%	9.95%	10.11%
Annualised Return (including costs)	-4.05%	13.57%	-1.68%	10.65%	33.31%	35.37%

Table 9: Trading performance - Final results

As can be seen from table 9, the GEP model continues to demonstrate a superior trading performance despite a significant high maximum drawdown. The GP strategy also performs well and presents the second highest annualised returns after transaction and leverage costs are considered. In general, the results show that leverage is marginally able to improve annualised return across all models. A further note worthy point relating to this

[11] Since most of the models have an in-sample volatility of about 20% this has been chosen as our target level. The leverage factors retained are given in table 6.

[12] 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. Hence in reality, inorder to calculate the interest costs 360 days per year should be utilised. But for the sake of simplicity, 252 trading days is utilised inorder 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.

significant trading performance is that the time required to train the GEP and the GP is only 15 minutes on a modern desktop personal computer.

7. CONCLUDING REMARKS

This paper applies the GEP and GP algorithms to generating a one-day-ahead forecast of the ASE 20 fixing series, using only autoregressive terms as inputs. In addition, as benchmarks, a MACD, an ARMA and a MLP model are utilised benchmarks. The in-sample period under consideration is January 2001 - August 2007 and the models out-of-sample trading efficiency is evaluated over the period September 2007 to December 2008.

The GEP algorithm demonstrates a higher trading performance when measured using annualised return and information ratio (before transaction costs). When refined trading strategies are applied, and transaction costs are considered, the GEP algorithm continues to outperform all other models achieving the highest annualised return. The GP algorithm model also performs well and provides profitable forecasts even though only autoregressive series are only used as inputs. The results show that both models are able to capture the nonlinearities and the high complexity of the ASE 20 fixing series. This can be attributed to their unique architecture and their ability to address and quantify complex issues. Based on this ability, it can be argued that the GP and GEP algorithms would excel in any pattern recognition exercise irrespective the nature of the series under study and to provide more accurate forecasts compared to traditional linear statistical models, such as ARMA.

It is also important to note that the GP and GEP algorithms which present the best performance needs around 15 minutes training time with a modern personal computer making them usable in a real-life quantitative investment and trading environment. Consequently, our results should go some way towards convincing a growing number of quantitative fund managers to experiment beyond the bounds of traditional statistical models and GAs.

APPENDIX

A.1 ARMA Model

The output of the ARMA model used in this paper is presented in figure 9 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	

Fig. 9: The ARMA model

A.2 Performance Measures

The performance measures are calculated as in table 10:

Performance Measure	Description	
Annualised Return	$R^A = 252 * \frac{1}{N} \sum_{t=1}^{N} R_t$ *with* R_t *being the daily return*	[8]
Cumulative Return	$R^C = \sum_{t=1}^{N} R_t$	[9]
Annualised Volatility	$\sigma^A = \sqrt{252} * \sqrt{\frac{1}{N-1} * \sum_{t=1}^{N} \left(R_t - \overline{R}\right)^2}$	[10]
Information Ratio	$IR = \frac{R^A}{\sigma^A}$	[11]
Maximum Drawdown	*Maximum negative value of* $\sum \left(R_t\right)$ *over the period* $MD = \underset{i=1,\cdots,t;t=1,\cdots,N}{Min} \left(\sum_{j=i}^{t} R_j \right)$	[12]

Table 10: Trading simulation performance measures

REFERENCES

Alavi A.H., Aminian P., Gandomi A.H., Arab Esmaeili M.,(2011), 'Genetic-Based Modeling of Uplift Capacity of Suction Caissons', *Expert Systems With Applications,* 38, 10, 12608-12618.

Alavi A.H., Gandomi A.H.,(2011), 'A Robust Data Mining Approach for Formulation of Geotechnical Engineering Systems', *Engineering Computations,* 28, 3, 242-274

.

Almanza, A. and Tsang, E. (2007), 'Detection of stock price movements using chance discovery and genetic programming', *International Journal of Knowledge-Based and Intelligent Engineering Systems*, 11, 5, 329-344.

Alvarez-Díaz, M. and Alvarez, A. (2005), 'Genetic multi-model composite forecast for non-linear prediction of exchange rates', *Empirical Economics*, 30, 3, 643-663.

Bekiros, S. D. and Georgoutsos, D. A. (2008), 'Direction-of-change Forecasting Using a Volatility-Based Recurrent Neural Network', Journal of Forecasting, 27, 407–417.

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

Chen, S. (2002), *Genetic Algorithms and Genetic Programming in Computational Finance*, Kluwer Academic Publishers, Amsterdam.

Chiu, D. and Chen, P. (2009), 'Dynamically exploring internal mechanism of stock market by fuzzy-based support vector machines with high dimension input space and genetic algorithm', *Expert System with Application*, 1240-1248.

Cowles, A. (1933), 'Can stock market forecasters forecast?', *Econometrica*, 1, 309–324.

Dehuri, S. and Cho S. B., (2008), '*Multi-Objective Classification Rule Mining Using Gene Expression*', Third International Conference on Convergence and Hybrid Information**.**

Donaldson, R. G. and Kamstra, M. (1996), 'Forecast Combining with Neural Networks', *Journal of Forecasting*, 15, 49–61.

Divsalar M, Firouzabadi AK, Sadeghi M, Behrooz AH, Alavi AH.,(2011), ' Towards the prediction of business failure via computational intelligence techniques'*, Expert Systems* 28, 3, 209-226

Divsalar M, Roodsaz H, Vahdatinia F, Norouzzadeh G, Behrooz AH., (2012), 'A Robust Data-Mining Approach to Bankruptcy Prediction', *Journal of Forecasting,* 31, 504–523.

Ferreira, C., (2001), 'Gene Expression Programming: A New Adaptive Algorithm for Solving Problems', *Complex Systems*, 13, 87-129.

Ferreira, C. (2006), *Gene Expression Programming: Mathematical Modelling by an Artificial Intelligence*, Springer, 2nd edition, San Francisco**.**

Gandomi A.H., Yang X.S., Talatahari S., Alavi A.H.,(2013), 'Metaheuristic Applications in Structures and Infrastructures', Elsevier,Waltham, MA, USA.

Gandomi A.H., Alavi A.H.,(2011), 'Multi-Stage Genetic Programming: A New Strategy to Nonlinear System Modeling'*, Information Sciences*, 181, 23, 5227-5239.

Gandomi A.H., Alavi A.H., Mirzahosseini M.R., Moghadas Nejad F.,(2011), 'Nonlinear Genetic-Based Models for Prediction of Flow Number of Asphalt Mixtures', *Journal of Materials in Civil Engineering, ASCE,* 23, 3, 248–263.
Grosan, C., Abraham A. (2006), 'Stock Market Modeling Using Genetic Programming Ensembles', *Genetic Systems Programming*, 13, 131-146.

Grosan, C., Abraham A., Han S. Y., and Ramos V., (2005) 'Stock market prediction using multi expression programming,' in *Portuguese Conference on Artificial Intelligence, Workshop on Artificial Life and Evolutionary Algorithms* (A. C. C. Bento and G. Dias, eds.), vol. 13 of Studies in Computational Intelligence, pp. 73–78, IEEE Press.

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

Kaboudan, M. A., (2000), 'Genetic Programming Prediction of Stock Prices', *Computational Economics*, 16, 207-236.

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

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.

Kwon, Y., and Moon, B. (2003), 'Daily Stock Prediction Using Neuro-genetic Hybrids', *Genetic and Evolutionary Computation*, 2203- 2214.

Lee, Y. and Tong, L. (2011) 'Forecasting time series using a methodology based on autoregressive integrated moving average and genetic programming', *Knowledge-Based Systems*, 24, 1, 66-72.

Lopez, H. S. and Weinert, W. R. (2004), 'An Enhanced Gene Expression Programming Approach for Symbolic Regression Problems', *International Journal of Applied Mathematics in Computer Science*, 14, 375-384.

Margny, M. H. and El-Semman I. E., (2005), '*Extracting Logical Classification Rules with Expression Programming: Micro Array Case Study*', AIML 05 , Conference 19-21 December , Cairo, Egypt.

Matias, J. M. and Reboredo, J. C. (2012), 'Forecasting Performance of Nonlinear Models for Intraday Stock Returns', *Journal of Forecasting*, 31, 172–188.

Nag, A. and Mitra, A. (2002), 'Forecasting Daily Foreign Exchange Rates Using Genetically Optimized Neural Networks', *Journal of Forecasting*, 21, 7, 501-511.

Shapiro, A. F. (2000), 'A Hitchhiker's Guide to the Techniques of Adaptive Nonlinear Models', *Insurance, Mathematics and Economics*, 26, 2, 119-132.

Sermpinis G., Laws, J., Karathanasopoulos, A, and Dunis, C., (2012), 'Forecasting and Trading the EUR/USD Exchange Rate with Gene Expression and Psi Sigma Neural Networks', *Expert Systems with Applications*, 8865-8877.

Yaghouby F, Ayatollahi A, Bahramali R, Yaghouby M, Alavi AH., (2010), 'Towards automatic detection of atrial fibrillation: A hybrid computational approach', *Computers in Biology and Medicine* 40, 11, 919-930

Yang X.S., Gandomi A.H., Talatahari S., Alavi A.H.,(2012), 'Metaheuristics in Water Resources, Geotechnical and Transportation Engineering', Elsevier,Waltham, MA, USA.

Zargari S.A., Zabihi S., Alavi A.H., Gandomi A.H., (2012), 'A Computational Intelligence Based Approach for Short-Term Traffic Flow Prediction', *Expert Systems,* 29, 2, 124–142.