*University College London*

# Dynamic Optimisation of Technical Trading Rules Using Genetic Programming

## MSc Intelligent Systems Project 2003/2004

Tom Graham

Project Supervisor: Chris Clack

---

This report is submitted as part requirement for the MSc Intelligent systems degree in the Department of Computer Science at University College London. It is substantially the result of my own work except where explicitly indicated in the text. This report may be freely copied and distributed provided the source is explicitly acknowledged.

---

**Abstract**

This Report presents the design of a GP-based system for learning technical trading rules for financial markets. These rules are generated from combinations of commonly used technical indicators. The design includes a novel dynamically adaptive learning algorithm for continuously evaluating and updating the GP candidate solution population according to new market data. The trading performance of the system was tested in a series of experiments using historical price data from individual stocks and from the Dow Jones Industrial Average. Results show that the GP-generated trading rules were unable to consistently produce excess trading profits, relative to a simple buy-and-hold strategy, using either the standard 'static' GP learning algorithm or the new dynamic method.

# 1. Introduction

Technical analysis is a technique widely used by investment managers and financial market traders for predicting future movements of asset prices. There is already a large body of work which investigates the use of Genetic Programming (GP) methods to learn profitable trading rules from combinations of technical indicator functions. Two main points emerge from the available literature. Firstly, the usefulness of technical trading rules generated in this way has not been proven: The ability of these rules to produce a trading profit that is significantly in excess of a relevant benchmark has not been demonstrated conclusively when used in simulated trading with out-of-sample data. Secondly, those authors who do claim significant excess trading returns note that this effect disappears as the input data gets further away in time from the data used to train the GP system. This suggests that market conditions change over time in such a way as to reduce the predictive power of GP-generated trading rules.

This work investigates the ability of GP-generated technical trading rules to produce profitable returns from simulated trading for individual stocks and for the Dow-Jones Industrial Average. The design of the GP-based system for rule optimisation and trading takes into account the implementation details from previous similar experiments, as described in the available literature, together with the author's own adaptations. Details of the system design are given later in this report together with a discussion of design choices and of issues arising from the implementation.

In initial experiments the GP system was trained using a fixed sequence of historical asset price data and the evolved trading rules were evaluated in simulated trading on subsequent ranges of data. Particular focus is given to out-of-sample trading performance of the GP-generated technical trading rules relative to simple 'naïve' trading rules. Various methods for limiting the effect of over-fitting of the GP-evolved rules to the training data were tested, with the aim of maximising out-of-sample performance. This work includes an

investigation of the functional behaviour of the trading rules resulting from training of the GP system on particular sequences of price data.

Further experiments were carried out with a system that uses periodic retraining to generate a sequence of optimised trading rules, and then with a system which implements a novel dynamic learning GP algorithm designed to continuously train and adapt to new data. Both of these methods were used to address the issue of the degradation of performance of individual optimised trading rules over time. One aim of these experiments was to investigate the nature of the changes in market behaviour which affect trading rule performance: The question of whether these changes are incremental, or whether they occur in discrete steps, was addressed by analysis of the changes in the structure and fitness of evolved trading rules over time. The dynamic GP algorithm was specifically designed to overcome problems relating to population convergence which have been seen to impair the performance of evolutionary techniques in applications which require dynamically adaptive learning.

This report begins with an introduction to the theory and practice of technical analysis and a description of evolutionary computation techniques, with a particular focus on Genetic Programming. This is followed by a review of previous work on the application of Genetic Programming to the field of asset price forecasting and trading, and an analysis of techniques for implementing dynamically adaptive learning systems. Details of the design and implementation of the GP system are included next, together with a discussion of various design choices and also a description of the dynamically-adaptive GP algorithm. The next section gives a statement of the goals of the experimental work and describes the experimental procedures for achieving these, followed by the presentation and analysis of experimental results. Concluding remarks and suggestions for further work make up the final section.

# 2. Background

## 2.1 Technical Analysis

Technical Analysis uses the historical price data of traded assets in order to identify patterns or trends in the price movement. These are used to predict probable future trends, and thus indicate whether to buy or sell the asset.

Particular classes of pattern are identified using Technical Indicators (TI). These indicators can be qualitative - looking for recognisable shapes in a plot of time series data - or quantitative. Many TI are derived from moving average calculations, or from sequences of local minima and maxima. A simple example of the latter is the Price Channel Breakout (PCB) indicator. In this case the current price is compared to the local minimum and maximum values within a given time widow. If the price goes outside this range the trend is predicted to carry on in the same direction.

### 2.1.1 Theoretical basis for technical analysis

This approach is based on the belief that asset price time series exhibit certain regularities. These regularities are explained in terms of psychological factors affecting the group behaviour of investors in the market. An example of this behaviour is used to justify the use of the PCB technical indicator: If the price of an asset has dropped from a previous peak, investors will think they have lost out by failing to sell at this peak value. When the price next reaches the same value they will be inclined to sell, inducing another price drop. This will happen repeatedly until the ‘resistance line’ is broken at which point the price is expected to continue on an upward trend.

### 2.1.2 Technical Analysis and the Efficient Market Theory

This supposition is contradicted by the Efficient Market Hypothesis (EMH), which states that all information about an asset’s value - including historical price information - is reflected in the current price (see Malkial, 1999). According to this theory, any regularity in price movement will be detected by

‘the market’. This information will affect trading in such a way as to destroy the regularity. Market prices follow a random walk and so are essentially unpredictable in nature.

A weaker formulation of the EMH states that, although all markets may not be perfectly efficient at all times, inefficiencies can not be exploited to make a trading profit - any profits made will be negated by transaction costs. Because Technical analysis depends on inefficiency in the market, any evidence that this technique works can be seen as evidence against the EMH.

The debate about the validity of the EMH and the effectiveness of Technical Analysis has not been resolved at present; the work of Alexander (1961) and Fama (1970) finds no evidence for the effectiveness of TI in making profitable trading decisions, concluding that Technical Analysis is a pointless practice. However, there is some work that contradicts these findings (Levich & Thomas, 1993), and many investment professionals trading in stocks, commodities and currencies use Technical Analysis methods.

## 2.2 Evolutionary Computing

The field of Evolutionary Computation is concerned with using computer programs to emulate the mechanisms of natural selection for the purpose of optimisation, adaptation or search. All evolutionary algorithms involve the representation of set of possible solutions to a given problem as a population of individuals. The performance, or fitness, of each candidate solution is tested and the best individuals are permitted to survive and produce ‘children’ based upon themselves. This is analogous to Darwinian selection, which is seen to create complex and highly adapted organisms - optimised solutions to the problem of survival and reproduction in the natural environment. Particular evolutionary algorithms differ in the way candidate solutions are represented and how they are reproduced.

One advantage of Evolutionary Computing techniques is that they can be used on non-numerical problem domains. Many other optimisation methods such as hill climbing algorithms and artificial neural networks rely on a numerically differentiable solution space. Evolutionary algorithms are an efficient way of searching for optimum solutions due to the effect of 'parallel search': Many candidate solutions are evaluated simultaneously, and high fitness individuals are recombined to create new solutions, so a wide region of the search space can be explored efficiently.

### 2.2.1 Genetic Algorithms (GA)

A widely used Evolutionary Computing technique is the Genetic Algorithm (GA), originally developed by Holland (1975). A Genetic Algorithm operates on a population of individuals represented by character strings. These are evaluated according to a fitness function appropriate to the problem in hand. Pairs of individuals, selected at random but biased according to fitness, are recombined to create members of a new population. Starting from an initial population of randomly generated candidate solutions, successive generations are produced until some termination criterion is reached: This may be the convergence of the average and maximum fitness values, or simply a limit on the number of generations.

#### String representation

Genetic Algorithms represent candidate solutions as strings - finite sequences of characters from a given alphabet (typically binary or integer numeric). The method of mapping a candidate solution to a GA string depends on the problem domain: The string may represent, for example, an ordered sequence of operations, or a set of independent parameters. However, a particular location in the string sequence always represents the same part or parameter of the solution.

#### Crossover and mutation operations

The string representation used in GA is analogous to the structure of biological genetic material - DNA. In the same way, the method of creating new GA strings mimics the recombination mechanisms of DNA.

Crossover is the operation of exchanging information, or ‘genetic material’, between two individuals. It works by swapping the values at corresponding locations between pairs of strings. There are various methods for implementing crossover suited to different applications. The simplest method is single point crossover: a point is selected at random to divide each string into two sections, one of which is swapped over. Alternatively, a greater number of crossover points may be used so that more than one contiguous sub-sequence is exchanged between ‘parent’ strings. Another method, uniform crossover, acts on individual locations - swapping each according to a fixed probability.

The mechanism of crossover is believed to be the source of GA’s power as an optimising search method. This assumes that an optimal solution will be composed of individually optimal ‘genes’, where a gene can be thought of as a set of one or more position-value pairs from an individual string. A useful gene set will be passed on to an offspring only if all of its members remain together after crossover. The larger the set, the less likely it is to survive reproduction intact. The action of fitness selective bias in reproduction means that beneficial genes will tend to propagate through the population over time. Because of this, the diversity of different gene sets decreases and population members become more similar - on average - over successive generations. As the population converges in this way, it becomes more likely that the information being exchanged in crossover will be the same in both individuals. This increases the probability that genes will be reproduced intact; crossover any part of two identical strings and they will still be the same. The result is that the GA algorithm is able to combine progressively larger sets of optimal genes with successive generations - this is the Building Block Hypothesis of Genetic Algorithms (Goldberg, 1989).

Mutation is simply the action of randomly changing the value of individual locations or sub-strings within a GA sequence. Although crossover is the main factor in the evolutionary behaviour in GA, mutation is important because it is the only way of introducing new genetic material into the overall population.

**Fitness based selection**

As stated previously, individuals are selected for reproduction randomly, but with the probability of selection weighted according the measured fitness of the candidate solution. There are three commonly used methods to implement fitness based selection:

*Fitness Proportional Selection (FPS)*

The sum total of the fitness values of all population members is calculated and a random number is selected between zero and this value. Running through all population members, the fitness values are summed a second time. When the sum exceeds the randomly generated number, the current population member is returned. If the total fitness sum is thought of as the circumference of a circle, then each individual is represented by a sector of the circle equal to its fitness value. If a pointer is placed at a random position on the wheel, the probability of it falling within any individual sector is proportional to the fitness of that individual. This is why FPS is also known as 'Roulette Wheel Selection'. A disadvantage of this method is that a fitness proportional selection weighting may not always be suitable. It may be desirable to disproportionately bias selection in favour of individuals whose fitness is only marginally greater than average, or to have only a small bias towards individuals who have very high relative measured fitness. Another problem with this method is that it does not work with negative fitness values.

*Rank Selection*

This method works like FPS, only the fraction of the 'roulette wheel' assigned to each individual is dependent on rank position rather than absolute fitness. The degree of bias can be controlled by using the rank position value raised by a chosen polynomial factor. This is a comparatively slow method because the population must be sorted according to fitness.

*Tournament Selection*

A group of Individuals are selected from the population at random. The fittest member of this group is returned. The degree of selection bias is related to the size of the group or tournament - the greater the size, the greater the relative weight of fitter individuals: With a tournament size of two, the fittest member of the population is twice as likely to be selected as the median. This method is the most computationally efficient as only the individuals selected for the tournament need to be inspected.

### 2.2.2 Genetic Programming (GP)

GP is an extension of GA, originated by Koza (1992). GP uses a similar evolutionary procedure for search and optimisation based on selective recombination from a population of candidate solutions. It differs from GA in the representation of the candidate solutions.

**Tree structure representation**

The tree structure is a hierarchical graphical model consisting of a set of interconnected nodes (see figure 1). Each node can have several connections to nodes at a lower level, but only a single ‘parent’ connection.

The name Genetic Programming refers to the fact that the tree structure is usually used to represent a function in the style of a computer program syntax tree. The branch nodes represent functions - they take values passed by their immediate descendents as input arguments and return an output to their parent. The terminal ‘leaf’ nodes represent input arguments or variables. In this way the branching hierarchy denotes the nesting, or evaluation ordering of functions. The tree structure can be used in other ways according to how a particular problem domain is best represented. For example Li & Tsang (1999) use the GP structure as a decision tree.

In contrast to Genetic Algorithms, the tree representation of GP is able to generate candidate solutions of variable size and complexity - crossover and mutation operations can alter the size of individual trees. Another important

difference is that, unlike GA, there is no specific mapping of individual parts of the tree to a part of a candidate solution. The GP function parse tree returns a single output value from a given set of input variables.

**Operators and terminals**

The GP tree structure is constructed from two sets of node types - functions and terminals. The branch nodes - those which have at least one connection to a child node - are taken from the function set. This set typically consists of simple logical (AND, OR, etc), conditional (IF-THEN-ELSE) arithmetic (+, -, *, /), or comparison (<, >, =) operators. The choice of function set is a design decision which depends on the problem domain and on the data types that GP function should take as input and return as output. The terminal set consists of all the data input variables which are to be evaluated by the GP function.

Function and terminal sets must be chosen such that they are capable of expressing a solution to the problem. This means that the designer should have knowledge about the problem domain - including some idea of the likely form of solutions.

**GP Crossover and mutation**

Genetic Programming implements crossover and mutation operations equivalent to those used in GA. To carry out crossover on a pair of GP trees, a single node is selected at random from each - these form the crossover points. The sub-trees originating at these nodes are swapped over, creating two new GP trees (shown in figure 2). If the two sub-trees contain a different number of nodes then the resulting offspring trees will be of different sizes to the parents. Crossover is easy to implement in code by swapping over pointers between parent and child nodes at the selected points.

Mutation works in a similar way - a new randomly generated sub-tree is inserted at a randomly selected node and the displaced section is discarded. Because crossover can exchange sub-trees between different locations, unlike in GA, there is less need for mutation in creating and maintaining

diversity in the population of candidate solutions. Therefore the mutation operator is sometimes left out of GP algorithms if the population is made large enough to ensure sufficient initial diversity of available building blocks (Mitchell, 1998).

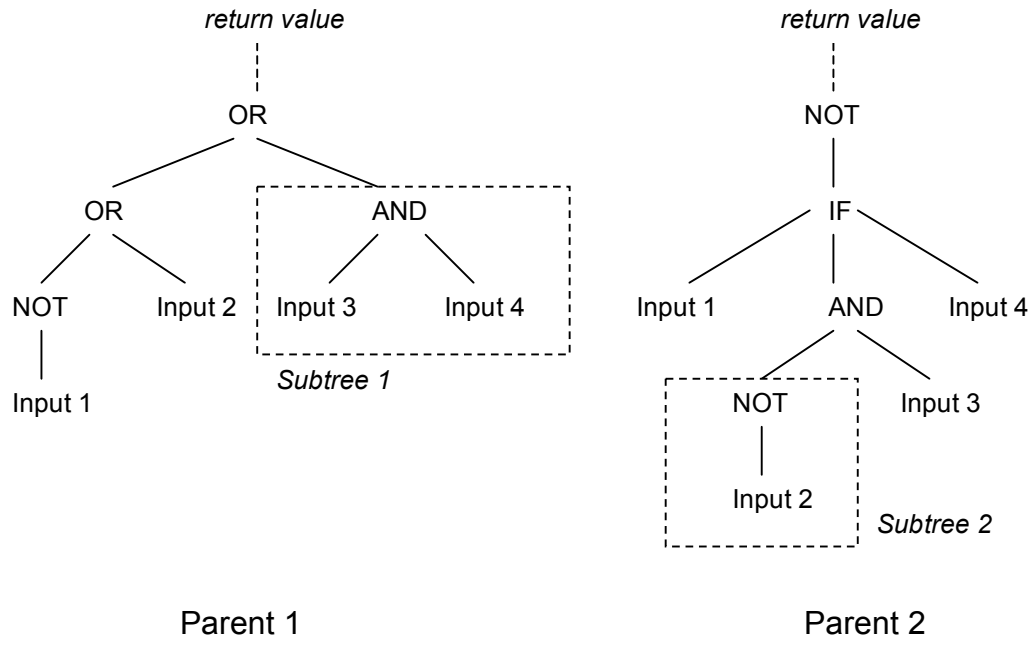


*Figure 1: GP parse-tree representation of two functions taking four separate input parameters*

.

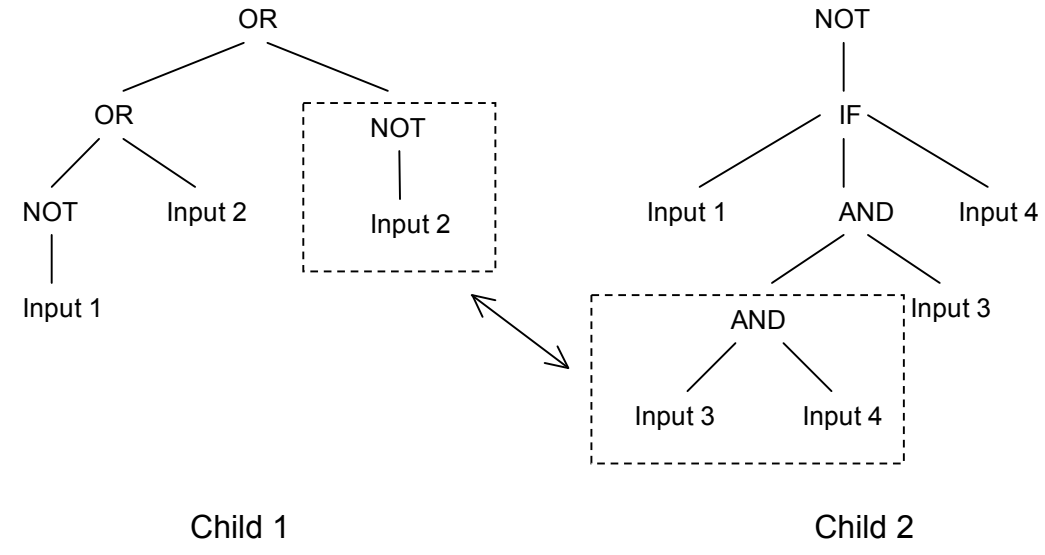


*Figure 2: Result GP of crossover.*

## 2.3 Application of GP to technical analysis

Research indicates that the application of individual TI to asset price forecasting does not result in significant improvements in investment returns (see Fama, 1970; Dempster and Jones, 2000). It is common practice for traders in financial markets to combine individual TI into trading rules. The problem of learning profitable trading rules can be seen as a search for optimal combinations from a finite set of base mathematical functions. This problem domain would seem to be well suited to the GP representation of candidate solutions. There already exists a significant body of work which addresses the application of GP in this way (see following section).

### 2.3.1 Literature review

The following summary and analysis of relevant literature serves two purposes: Firstly, the implementation details of previous GP trading systems are assessed and compared. This information is used to assist in the design of the GP system used in the current project. Secondly, the results reported for each method are analysed in order to ascertain whether there is valid evidence of profitable trading performance for any of the systems. The last paper included in the literature review does not relate specifically to the application of evolutionary computing methods to market forecasting or trading - rather it is concerned with the general effectiveness of evolutionary algorithms in dynamic optimisation applications.

***Using Genetic Algorithms to find technical trading rules,***
***Franklin Allen and Risto Karjalainen (1999)***

The authors used genetic programming to learn technical trading rules for the S&P 500 index. The GP system creates trading rules using a set of simple input functions (minimum, maximum and average) computed from the historical index price data. The functions take a numerical value input which specifies the time window over which to evaluate the function. They output a

numerical value. The terminal node set consists of these functions plus the current price value and some numerical constants.

The function set for this GP system uses a combination of logical operators (AND, OR, etc), arithmetic operators (+, -, *, /), comparison operators (<, >, =) and numerical and Boolean constants. Because the non-terminal nodes operate on a mixture of Boolean and numerical data types the crossover and mutation functions have to implement a method of type checking to ensure that sub-trees of nodes are only substituted with those which return the same data type.

The report includes particularly detailed statistical analysis of the results. The authors conclude with the following assertion:

> *"After transaction costs, the [GP-evolved] do not earn consistent excess returns over a simple buy and hold strategy".*

***Optimization of Technical Rules on the Basis of Intelligent Hybrid Systems,***
***A. Kapishnikov (2002)***

The GP system described here uses a set of existing commonly used mathematical Technical Indicator functions - including Moving Average Convergence-Divergence (MACD) and K-Stochastic - for the terminal node inputs. Boolean function nodes (AND, OR, XOR etc) are used to combine the input TI values into a single output function. The method of constructing trading rules from combinations of known TI is an alternative to that used by *Allen and Karjalainen* which evolves rules from a set of simple mathematical functions.

One unique aspect of this work is that technical indicators are combined with artificial neural network inputs in the terminal set for the GP trading system. Apart from the use of neural networks there are two other noteworthy features of the system design and experimental methodology:

The logical operators in the function set use three-way Boolean values, BUY/ SELL/HOLD, to match the output of individual TI, instead of TRUE/FALSE, which is a neat way of avoiding having to evolve separate GP rules for buy and sell indicators.

During training, transaction costs (the costs incurred from executing transactions in simulated trading) are set to a higher rate to minimize the influence of noise in the data and hence avoid over-fitting. In this way evolutionary selection may be biased towards rules which correctly predict large price movements but trade at a lower frequency and hence incur lower overall transaction costs.

Results showed that trading rules based on optimised combinations of Technical Indictors outperformed individual TI but did not significantly outperform benchmark buy-and-hold strategies on out of sample data when taking transaction costs into account. The performance of evolved trading rules on out-of-sample data was found to be higher the closer in time it was to the training data. This leads the author to suggest that implementation of adaptive learning may improve the results of the system.

***Improving Technical Analysis Predictions: An Application of Genetic Programming,***
***Jin Li and Edward Tsang (1999)***

This paper describes another system which evolves Genetic Programming based trading rules using technical analysis. Trading rules are created by combining a set of existing TI rules in the manner of *Kapishnikov*, rather than building combinations of simple functions with variable parameters as in *Allen & Karjalainen*.

The unique aspect of this work is in the GP representation of the trading rules. In all other cases seen, the GP tree structure represents a LISP-style function, where terminal nodes represent input variables and non-terminal nodes are operators (logical or arithmetic) which take values passed by their child nodes as inputs. Here, the GP structure represents a decision tree, where (non-

terminal) branch nodes evaluate a particular condition (e.g. “Is today’s price higher than the 10-day maximum”) and ‘leaf nodes’ are simply BUY, SELL or HOLD decisions. This results in trading rules in the form of IF(…) THEN(…) ELSE(…) statements.

This method is less flexible than the LISP style tree as it would not support the function set described in *Allen & Karjalainen* which uses a mixed set of numerical and logical operators and inputs. The decision tree method has exactly the same expressive power as a system using only logical operators with logical-valued TI inputs. Given that they have the same set of TI available as input you can represent the same conditional technical trading rule by either method using a similar number of nodes. This means that the two methods are functionally equivalent.

The authors report that the evolved trading rules can generate large profits. These claims may not be valid, because the simulated trading does not include any transaction cost factor which would be expected to reduce trading profits in real-life situations. Also the performance of the trading rules is not compared to that of a buy-and-hold strategy over the same time period. Therefore the results are not conclusive.

***GP-evolved Technical Trading Rules Can Outperform Buy and Hold, Lee A. Becker and Mukund Seshadri (2003)***

This paper presents experiments using GP-evolved technical trading rules, based on known TI functions, for forecasting the S&P 500 index. The one interesting methodology change from previous work is in the use of a fitness function which penalises complex trading rules. The purpose of this was to avoid over-fitting of the evolved trading rules in order to improve out-of-sample trading returns.

The authors report that without the complexity penalising factor the maximum-fitness evolved trading rules produced negative excess returns, relative to a baseline buy-and-hold strategy, in the out-of-sample evaluation period. When using the C-P factor there was a reduction in the evolved fitness values and

an increase in the average return for the out-of-sample period. This average value exceeds the buy-and-hold return for the same period - hence the title of the paper. However, the published data shows that the margin of difference is well within the Standard Deviation of returns over the range of GP runs, so the reported positive excess return does not appear to be statistically significant.

***A Real-Time Adaptive Trading System Using Genetic Programming, M. A. H. Dempster and C.M. Jones (2001)***

In this paper the authors assert two important points regarding Technical Analysis based trading systems. The first is that the use of individual technical indicators is not effective for making profitable trading decisions. Therefore it is common practice to use technical trading rules which consist of combinations of a range of indicators. The second point is that in the real world, particular trading rules are only used for a finite period of time: as market conditions change a trading rule which was previously found to be profitable will become loss making. Traders must therefore find new trading rules over time in order to continue to make excess returns on the market.

In the experiment described, the GP evolved strategies were prompted to re-optimise when trading losses were encountered. It is not entirely clear how the re-optimisation takes place but it seems to be by retraining the GP from scratch with the latest quarter's market data. The results show that static evolved strategies perform worse the further in time the system runs from the period of the training data; this suggests re-optimisation would be beneficial. However, in further experiments where periods of loss trigger re-optimisation, overall performance actually deteriorates. The explanation offered for this is that in retraining the strategies are over-fitted to the latest period of data only and is in effect "chasing losses".

The authors' assertion that statically evolved trading rules are "significantly profitable" is somewhat dubious as they assume that any positive return from trading two currencies on the Foreign Exchange (FX) market is significant. They do not account for the fact that long term trends in exchange rates may

make a simple buy-and-hold strategy profitable over the time period from which they take both their training and validation data.

The paper is particularly relevant to this project because the system implements GP based adaptive selection of technical trading rules, although the system retrains from scratch rather than implementing a true dynamic (i.e. continuously adaptive) optimisation method. The poor experimental performance of this method suggests that this is a suitable area for further investigation.

An appendix gives mathematical descriptions of a number of Technical Indicators, including Simple Moving Average Crossover (SMAC), Adaptive Moving Averages (AMA), Price Channel Breakout (PCB), K-Stochastic, Relative Strength Index (RSI), and the Commodity Channel Index. Several of the TI functions used in the system program for this project are based on these definitions.

***Diversity Does Not Necessarily Imply Adaptability,***
***Marcus Andrews and Andrew Tuson (2003)***

Evolutionary techniques are by nature well suited to dynamic optimisation problems as they mimic natural processes which are inherently dynamic. A diverse population is desirable for dynamic optimisation problems as it allows continuous exploration of a search space.

This paper compares the performance of Evolutionary Algorithms (in this case a GA) to a standard hill-climbing algorithm on a simple test dynamic problem. The results show that the performance of the GA is generally superior. The authors also investigated the effect of population size on performance of the GA. They state that population size has “a complex effect on performance when adapting to changing problems”.

### 2.3.2 Summary

None of the work presented in the preceding section gives a clear demonstration of the ability of GP evolved technical trading rules to

consistently generate significant positive excess returns in realistically simulated trading. Several of the authors do make such claims but careful analysis of the published data shows that their results do not back these up conclusively. The report which contains the most rigorous statistical analysis of experimental results (Allen and Karjalainen, 1999) reports no significant excess returns.

## 2.4 Dynamic learning techniques

The work of Dempster and Jones (2000) and Kapishnikov (2002), described above, indicates that technical trading rules which are optimal (in the sense of generating trading profits) in one particular period will subsequently lose efficacy over time. These results suggest that it may be beneficial to implement some form of dynamic learning in the system which generates trading rules. Such a system should be able to adapt the rules it produces over time in order to maintain optimal trading performance.

### 2.4.1 Dynamic application of evolutionary computing

Previous work (e.g. Andrews and Tuson, 2003) suggests that learning systems based on evolutionary computing methods can perform poorly on dynamic optimisation problems. The term ‘dynamic optimisation’ refers to a problem where the optimum solution is not always the same, or where the problem domain evolves in some way over time. The poor performance of evolutionary systems in these situations is attributed to *population convergence*: because high fitness members are favoured in reproductive selection, they and their offspring tend to dominate the population over successive generations of breeding. This results in a population which is eventually filled with copies of the fittest members - leading to a lack of genetic diversity. The result of this is that when there is a change in context or environment it is difficult or impossible for new solutions to be created from the available genetic stock.

Andrews and Tuson suggest that greater population diversity does not necessarily lead to improved adaptability in evolutionary systems. However, they equate diversity solely with population size, which is incorrect. It is reasonable to say that a large random population will *initially* be more diverse than a small one as it contains a greater number of candidate solutions. However, experiments with static population evolution show that convergence occurs no matter what the initial population size. This is a natural result of selection pressure which is a fundamental part of any GA or GP algorithm. A large population of identical (or nearly identical) genotypes is no more diverse than a small one.

Population convergence is perhaps less of an issue for GP than GA because in the former it is possible to create distinct new structures from two identical parents under crossover (by moving sub-trees between different locations), whereas in the latter this is not the case. Convergence can still be a problem in that it may cause individual operator or terminal node types to be eliminated entirely from the population, in which case the mutation operation becomes important.

Another potential advantage of GP over GA in dynamic applications is the possible existence of ‘introns’ in GP population members. The term ‘intron’ comes from biology and it refers to the existence of non-functional genetic code in chromosomes. It has been observed that a significant proportion of the genetic material in, for example, the human genome is never expressed - i.e. particular genes are not used biologically. Introns are also observed in evolved GP populations in the form of sub-trees of operators and terminals that have no effect on the evaluation behaviour of individual parse-tree functions. The proportion of intron material in GP trees is often observed to increase with successive evolved generations, a phenomenon known as ‘GP bloat’. The explanation for this is that a good candidate solution which contains a high proportion of introns and a low proportion of active genetic material is less likely to lose beneficial active material under crossover, and hence is more likely to produce high fitness children. The effective fitness of individuals thus depends not only on how it is scored by the fitness function,

but also on its ability to pass on useful characteristics to subsequent generations. Introns benefit reproductive fitness and therefore tend to propagate through the population over time even though, by definition, they have no direct effect on the evaluated fitness of individual candidate solutions. The presence of redundant material in a GP population may improve its ability to adapt dynamically, because it may include building blocks of operators and terminals which will be beneficial to candidate solutions at some future time.

### 2.4.2 Other methods

A popular method for dynamically adaptive learning is *Swarming* (Eberhart & Shi, 2001). This is a population-based method using a fixed-sized set of candidate solutions. In contrast to evolutionary computing, individual population members move through the solution space over time. Population dynamics are governed by a simple set of rules in such a way that the population is 'attracted' to the fittest member at any given time (hence the name). If the maximum fitness region of the search space moves over time the motion of individuals around the previous centre should allow this movement to be tracked. Convergence is avoided by the use of a rule which prevents individuals from moving within a certain distance of each other. Unfortunately it is not practical to implement this method using the GP parse-tree representation of candidate solutions as there is no clear concept of relative position or movement in solution space for such individuals.

Another candidate method for dynamic learning is that known as *Expert Tracking*, described in Herbster and Warmuth (1998), which is a dynamic adaptation of the fixed *Expert Share* algorithm. This algorithm uses a predefined fixed population of candidate solutions or 'experts' and learning occurs in a series of trials. The system predicts the outcome of each trial according to a weighted sum of the predictions of each individual expert. At each step the weight assigned to the experts is updated according to their accuracy in predicting the previous trial. Individuals that consistently give accurate predictions receive exponentially increasing weighting over time. The problem with this method is that poorly performing experts can attain arbitrarily small weightings which will not be recovered at some future time

when they become the best predicting experts: this can be seen as another form of convergence. The *Expert Tracking* algorithm alters the weight-updating formula to place a lower limit on the relative weighting of individual experts so that small weights can be recovered. This framework could be used in the setting of predicting future values of asset prices, however the fact that it uses a fixed set of experts - new candidate solutions cannot be generated - makes it less than ideal for this particular application.

# 3 System Design and Implementation

## 3.1 Problem Definition

The material goal of this project is to produce a system for the GP-based optimisation of technical trading rules using historical asset price training data. This system must be able to perform the standard ‘static’ method for evolving optimised candidate solutions, as described in section 2.2 and in Koza (1992), and also implement an appropriate dynamically adaptive GP learning algorithm.

In order both to train the GP system and to evaluate individual trading rules it is necessary to implement a method of simulated asset trading which accurately represents how the rules would perform in real life. This will involve firstly calculating TI values from the input market data, then evaluating the trading rules based on these values, and then enacting the trading recommendations and recording resulting profits for each individual candidate solution.

The design for the system should take into account all useful information gained from previous work conducted in this field. The works covered in section 2.3.1 describe various measures to reduce the effect of over-fitting of the evolved rules to the training data, or to otherwise improve the trading performance of the GP-generated rules. In order to develop a system which achieves the best possible performance it is desirable to combine all of the most promising of these techniques into the design.

The system should be as adaptable as possible, in terms of the control parameters for the GP algorithm and in other aspects of the design, in order to allow maximum optimisation of system performance.

## 3.2 Development Tools

The system program was initially written in the Java programming language using the Sun Java compiler and virtual machine. The Java language was chosen because the design required an Object Oriented Programming language. Also, Java allows easy handling of pointers and automated garbage collection and there are features particular to this language which made certain parts of the design easier to implement. Later the program code was ported to J# (essentially the same language with different class libraries) in order to run on the Microsoft .NET platform and this was found to deliver a significant increase in speed performance.

## 3.3 Implementation details

### 3.3.1 GP tree structure representation of trading rules

The GP-based technical trading systems describe in section 2.3 can be divided into two categories, according to the way trading rules are constructed within the GP tree structure.

The first group, exemplified by Allen and Karjalainen (1999) and Jonsson et al (1997) uses just simple min/max and moving average functions, the current price value and numerical constants as the input set. These are combined using a large operator set including arithmetic, conditional and logical functions. The second group uses a set of known TI functions, such as PCB and Relative Strength Index (RSI), which are combined using Boolean operators.

The advantage of the first method - letting strategies evolve from simple functions - is that, in a certain sense, it allows a greater degree of optimisation of the trading rules. The use of arithmetic and comparison operators allows rules to be created that could not be expressed using only Boolean functions. Also, having the min/max and average functions take a numerical input that defines the evaluation time window creates an extra level of optimisation.

Practitioners of technical analysis do adapt moving average based trading rules in this way to model supposed variations in the period of oscillating price trends.

Other TI which have more complex mathematical formulations, such as the RSI, also have parameters which, technically, could be optimised within the GP in this way. It would be impractical to implement this however, as these TI are functions of their own previous values: therefore allowing variable parameters in these cases would necessitate the recalculated of an entire sequence of TI values for every occurrence of that TI in every GP-tree in the population at each time step.

To illustrate this point, several TI use Exponential Moving Average (EMA) calculations of the form:

$$EMA(T) = k \times price(T) + (1 - k) \times EMA(T - 1)$$

where T is the current time, k is a parameter between 0 and 1 and EMA(T - 1) is the previous EMA value.

The disadvantage with the method of *Allen and Karjalainen* is that it is limited - in another sense - by the simple input function set that it uses. Although many commonly used TI rules are derived from moving average calculations or local minima and maxima of past prices - and could therefore be generated from this simple function set - there are others which cannot be expressed in this way. The RSI indicator, for example, cannot be expressed as a combination of simple moving average and max/min functions.

Another problem is that it can be hard to evaluate how rules evolved in this way actually function; large, complex GP parse-tree functions can be hard to comprehend. This is a significant factor in the choice of implementation method as one of the aims of the experimental work was to investigate how

optimal trading rules function and how rules trained on different time-sequences of price data differ in their functional performance.

Taking these factors into account, it was decided to implement the second method - using a set of commonly used existing TI and combining them to form 'meta-indicator' rules. Another reason for this choice was that the alternate method was implemented by *Allen and Karjalainen* and, in the most rigorous experimental analysis of all the works described in section 2.3, it was found not to be effective.

### 3.3.2 Technical Indicator set

I have chosen the following set of Technical Indicators, based on the available financial literature, including Brock et al (1992), and on the TI sets used in previous experiments (e.g. Kapishnikov, 2002).

*Simple Moving Average*

If the current price crosses above the average price of the previous n days, return a BUY indicator; if the price crosses below return SELL.

*Price Channel Breakout (PCB)*

If the current price exceeds the maximum from the previous n days, BUY; if it goes below the minimum from this period, SELL; otherwise HOLD.

*Simple Moving Average Crossover (SMAC)*

If a short term (5-day) moving average value crosses above a long term (50-day) moving average then BUY; if the short term average crosses below then SELL.

*Moving Average Convergence Divergence (MACD)*

The MACD is the difference between a short term and long term price EMA values (as described in section 3.3.1). If the MACD crosses above its own EMA value, return a BUY indicator; SELL if it crosses below.

*Relative Strength Index (RSI)*

*K-Stochastic*

For a mathematical definition of these two indicators, see the appendix of Dempster and Jones (2001).

In addition to these indicators which are defined in the financial literature, the TI set also includes a *1-Day Price Change Indicator* which gives a BUY signal if the price has risen from the previous day's value and SELL if it has dropped. This is not a 'legitimate' TI as such: However, one of the naïve trading strategies that is being used to benchmark the performance of the GP-evolved trading rules simply predicts that the direction of price movement for the current day will be the same as on the previous day (This will be termed the 'Repeat Rule'). Therefore it seemed sensible to make the same rule available to the GP system.

### 3.3.3 Implementation of the GP structure for candidate solutions

The tree structure of candidate solutions is represented in the program as a set of instantiations of a node object class. There is a separate class for each function node type and for the terminal node, all derived from a base node class. The tree nodes are doubly linked; each has a pointer to its 'parent' node and the non-terminal function nodes have pointers to their 'children'. The function nodes are evaluated recursively so that the output value from the parse-tree is computed simply by evaluating the root node.

**Terminal node set**

The input node set consists of the following Technical Indicators; each can return the values 1, 0 and -1, representing BUY, HOLD and SELL recommendations respectively

1. 5-Day Simple Moving Average
2. 50-Day Simple Moving Average
3. 5-Day PCB
4. 50-Day PCB
5. SMAC

6. MACD
7. RSI
8. K-Stochastic
9. 1-Day Price Change Direction

Optionally, the terminal node set also includes three constant BUY, HOLD and SELL indicators.

In the implementation of the program, the price statistics on which the technical indicators are based (e.g. moving averages and various function values used in the evaluation of the MACD, K-Stochastic and RSI indicators) are calculated and stored in memory along with the price data itself. In this way the computation of the actual TI values at each time step only requires evaluation of simple conditional statements (e.g. IF [variable x(t) < variable y(t)] THEN Return 1). Furthermore, the representation of the terminal nodes within the GP tree structures uses pointers to the actual TI values so that these only have to calculated once at each time step, rather than every time an individual terminal node is evaluated. These measures allow fast and efficient evaluation of the trading rule output recommendations - the most time consuming part of the GP algorithm.

**Function set**

The function set consists of logical AND, OR and NOT plus the conditional IF operator. These are all modified to operate on the ternary BUY/SELL/HOLD logical values, as suggested by Kapashnikov (2002).

| AND | | |
|---|---|---|
| Input 1 | Input 2 | Output |
| B | B | B |
| B | H | H |
| B | S | H |
| H | B | H |
| H | H | H |
| H | S | H |

| | | |
|---|---|---|
| S | B | H |
| S | H | H |
| S | S | S |

| OR | | |
|---|---|---|
| Input 1 | Input 2 | Output |
| B | B | B |
| B | H | B |
| B | S | H |
| H | B | B |
| H | H | H |
| H | S | S |
| S | B | H |
| S | H | S |
| S | S | S |

| NOT | |
|---|---|
| Input | Output |
| B | S |
| H | H |
| S | B |

| IF | |
|---|---|
| Input (IF) | Output |
| B | Input(THEN) |
| H | H |
| S | Input(ELSE) |

*Table 1. Operator set truth tables*

### 3.3.4 Trading Strategy

The GP evolved trading rules will output a trading recommendation (BUY SELL or HOLD), at each time step, according to the current TI values and the structure of the individual rules. A trading strategy controls how these recommendations are acted upon in simulated trading of an asset.

The system can trade in either long or short positions. Trading long means a quantity of the asset is bought and held. A long position is exited when the total amount of the asset held is sold back into the market. Taking a short position involves selling a quantity of an asset that is not actually held by the seller. Short selling is permitted in some markets. The details of how short

trades are executed are complicated but essentially what happens is that a quantity of the asset is borrowed from a lending institution and immediately sold on the open market. Cash is provided to the lender as collateral, equal to the value of the borrowed assets plus an amount to cover possible trading losses (this collateral is equivalent to the investment capital used to buy long positions in the context of the trading strategy). A short trade is exited when an equal quantity of the asset is bought from the market to cover the original loan. There is often a limit on how long a short position can be held. The system is also allowed to take an out-of-market position, meaning no amount of the asset is currently held or sold short.

The system enters fixed-sized trades according to the trading rule recommendations. There is a time limit imposed on the holding of both long and short positions. Also there is a loss-trigger for exiting positions, such that if the value of an asset held in a long trade drops by a certain fraction from the original trade then the asset is sold. Alternately, a short position is exited if the value of the asset increases by the same fraction. In this way, once a position is entered there are three ways in which it may be exited, either an opposing trade recommendation is received, or the time limit is exceeded, or the loss trigger is activated. When a long position is held, a BUY or HOLD recommendation will cause no change. A SELL recommendation will exit the long position and immediately enter a short trade. Therefore, an out-of-market position will only occur when the time limit or loss trigger causes a long or short position to be exited, and will be held as long as the trade rule returns a HOLD recommendation.

This trading strategy is designed to reflect that which may be used by a Hedge Fund. Traditional pooled investment vehicles such as mutual funds are prohibited by regulations from trading in short positions. Hedge funds are legally structured in such a way as to avoid these and other regulatory constraints (see Connor and Woo, 2003). A trading strategy that can trade in short positions is able to make a profit from periods of negative market price change, whereas the best a long-only strategy can achieve during a downward trend is to stay out of the market.

A realistic representation of the trading strategies that are used in real life is necessary to validate any experimental results. There are some details of the implementation which only approximate certain aspects of a market trading system - for example in the way transaction costs are represented, and the fact that stocks can only be traded in discrete quantities is ignored. These measures are justified because the aim of the experiments is to compare the performance of GP-evolved trading rules relative to simple trading strategies, rather than to measure the absolute returns generated by these rules.

### 3.3.5 Imposition of transaction costs

Transaction costs are incurred whenever stocks are traded on an open market and these will affect the total profit or loss achieved by any trading system. The two main sources of transaction cost are transaction charges - the costs incurred when a trade is executed (e.g. brokerage fees) - and slippage - the loss resulting from differences between the price when the decision is made to enter a trade and the actual transaction price. This system models transaction costs using a simple percentage deduction from the value of each trade entered into.

### 3.3.6 Fitness Measurement

The fitness of individual technical trading rules is measured directly from the returns generated by simulated trading using those rules. The same fitness function calculation is used both to score candidate solutions for evolutionary optimisation, and for the out-of-sample evaluation of their performance.

Trading profit or loss is calculated daily according to the size and direction (long or short) of the current market position and the daily percentage price change in the asset value. Trades are always entered into in a fixed unit-size amount, minus transaction cost. Although returns are allowed to accumulate, or compound, for as long as a position is held, daily returns are relative to the initial investment amount. To illustrate this, suppose a long position is bought in a stock that appreciates at a constant 10% per day. If the transaction cost is 10%, deducted from the initial investment capital, the return for the first day

will be -1% (-10% deduction plus 10% growth on the remaining 90%) then +9.9% then +10.89% and so on…

The fitness function value is the daily return rate averaged over the evaluation period of simulated trading.

### 3.3.7 Dynamic Evolution Implementation Method

An effective dynamically-adaptive learning system based on evolutionary techniques depends on the preservation of diversity in the available candidate solutions in order to prevent population convergence. One way to achieve this aim is to employ active measures for maintaining genetic diversity in the population of candidate solutions. In order to do this, it is first necessary to have a way to measure diversity.

**Measurement of population diversity**

In this GP system, population diversity is indicated by the difference between the maximum and average fitness from the population. This is only an indirect measure of genetic diversity but it is easy to implement and relatively fast to compute. Convergence of the average fitness value to the maximum is used as an indicator to take action to increase diversity.

**Control of population diversity**

In the program there are two controls available to maintain or increase diversity over time. The first is the mutation rate - the frequency with which population members undergo mutation on their GP and GA structures: Mutation directly introduces more new genetic material into the population.

The second method is to control the selection pressure - the bias in selection of members for reproduction towards fitter members. Reducing this bias will reduce the rate of convergence within the population. This can be implemented using either the rank selection or tournament selection methods, as described previously. Both selection methods have been implemented in my system and can be selected by the user.

By adjusting the parameters controlling selection bias the system attempts limit convergence so that new trading rules can be generated. This is similar to the way in which the fixed-share weight update used by Herbster and Warmuth (1998) adapts the best expert tracking algorithm to limit the growth and decay of prediction weighting from a group of experts. The purpose of this work is also to allow dynamic adaptation to a changing problem context.

**Dynamic evolution of the population**

The standard (i.e. non-dynamic) method for evolving optimum solutions using GA or GP, which shall be referred to as static evolution, involves using a fixed set of training data to evaluate the performance of successive generations of candidate solution populations. The entire population is replaced at each generation by new members produced by fitness based selection, crossover and mutation from members of the old generation (except where elitism is used to carry over a proportion of the fittest members unchanged into the new generation).

In the case of market trading system the training data consists of asset price data covering a finite period of time. The fitness of candidate solutions is measured by running simulated trading over the period of the data and calculating the profit or return produced at the end of the run. All the work described in the literature implements this method in some way.

In order to create a dynamically adaptive system a new algorithm has been devised which measures the fitness of candidate solutions and selectively reproduces them continuously as successive day's price data are entered into the system.

It should be made clear that simulated trading is run for all the individual population members, using the latest available data, for the purpose of evaluating and evolving new trading rules. At the same time, simulated trading is run for the system as a whole, acting on the trading recommendations of the current designated fittest population member.

The algorithm works as follows:

***Initialisation***

1. Create an initial population of randomly generated GP trading rules. Optionally, run static evolution on this population, with a range of training data preceding the dataset used to run the main dynamic algorithm, in order to 'kick start' the development of optimal trading rules. Limit the number of generations to prevent convergence.

***At each time step***

2. Record the current price data; calculate TI values and record the profit earned for that day by each rule in the population. Evaluate each GP trading rule according to the TI values and update their market positions according to the BUY/SELL/HOLD recommendations and the trading strategy.
3. Evaluate the trading decision and record the daily profit for the system according to the current fittest trading rule.

***At every $i^{th}$ time step (where i is an adaptable parameter)***

4. Evaluate the fitness of all individuals over the preceding j days (j, the fitness range is also a variable). Select a small proportion of members to reproduce according to fitness.
5. Copy the selected parents, subject the children to crossover and mutation then retrospectively run simulated trading with them over the previous trading range - this is done so that the new individuals can be assigned a fitness value as soon as they have been created.
6. Temporarily remove the elite fittest members from the population. From the rest, randomly select a number, equal to the number of children, to replace. Selection in this case is biased according to inverse fitness - i.e. unprofitable trading rules are dropped from the population. Replace the elite members and new members into the population.
7. Record the average and maximum fitness values. Adjust the mutation rate and selection bias accordingly. Update the current designated fittest population member.

**Implementation of elitism for dynamic population optimisation**

Using the methods described above the aim is to maintain enough diversity to allow the system to adapt to changes in the environment - in this case changes in market conditions. In doing this the optimisation ability of the system may be impaired. An effective evolutionary learning system has to find the best trade-off of 'exploration versus exploitation'. Maintaining high diversity helps exploration - searching through the solution space for local maxima - at the expense of exploitation - optimising and using high fitness individuals to make trading decisions for the system. This makes the use of elitism especially important to ensure that a proportion of the fittest individuals are preserved at all times.

**Use of mutation**

Individuals in the population can undergo mutation, with a given probability, only at the point when they are created. It is not desirable to subject individuals to mutation once they are running trading evaluations on the data. This is because the fitness of individuals is based on trading performance measured over the preceding range of days within the fitness evaluation period. If an individual has undergone a mutation within this period the fitness results are invalidated. This may lead to the system making trading decisions, or creating new candidate solutions, based on individuals which performed well in the past but which have recently been 'disabled' by a harmful mutation.

## 3.4 System Testing

In order to ensure that the program is error free the following testing schedule was used. Every program class was tested independently to verify that it functioned as specified. The system program includes functions to output all data from the calculation of TI values and population fitness statistics for testing and evaluation purposes. Additionally the system can output text files which record the GP tree structure, fitness, trading frequency and TI weighting for individual candidate solutions.

**Test Schedule**

- *Verify correct calculation of TI from input data* - For simple TI the correct values can be calculated for comparison to the program output. More complex TI functions require published data for verification of output values.

- *Test data output to text files* - The program records and outputs both the statistics used to compute the Technical Indicator values (e.g. moving average values) and the actual TI output (1, 0 or -1).

- *Correct initialisation of GP trees* - Are the randomly generated trees structurally correct (and genuinely random)? Does each function node have the correct number of inputs? There is a facility in the program to output the structure of individual GP trees for examination.

- *Random selection of crossover and mutation points* - Ensure the method devised for selecting nodes at random from a pointer-linked tree structure works correctly.

- *Crossover and mutation functions* - Are the node sub-trees exchanged correctly?

- *Correct evaluation of GP parse-tree from TI inputs* - Given a set of input values, does the parse-tree return the correct output value according to the structure of logical operators?

- *Trading mechanism and calculation of daily profit* - Ensure that the system correctly implements the output recommendations according to the trading strategy.

- *Fitness calculation* - Does this function correctly?

- *Selection and basic evolutionary performance* - Verify that the average and maximum fitness values increase over time (see figure 3).

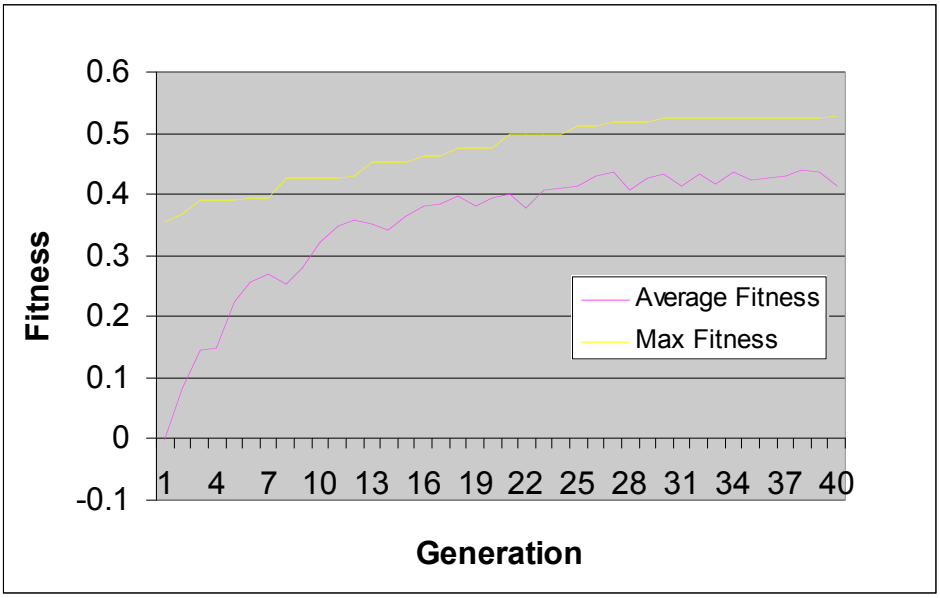


*Figure 3: Demonstrating the evolutionary performance of the GP system*

# 4 Experimental Design

The purpose of the experiments was to test the ability the GP system detailed in the previous chapter to generate profitable technical trading rules, and to compare the effectiveness of the static and dynamic learning methods.

## 4.1 Experimental Input Data

The experiments used stock market price data for simulated evaluation of trading strategies, both for the fitness evaluation of trading rules in the evolutionary stage of experiments, and for the out-of-sample testing of those rules.

All the stock market historical price data used for these experiments was obtained from the Yahoo Finance website. This is a respected source of accurate financial data used by academics and professionals. Yahoo provides data on all individual company stocks traded on major exchanges. Opening, closing, range high and low prices and trading volumes are given for each day. The trading system uses only the adjusted closing price data; this is the daily closing price adjusted for stock splits and dividend payments which otherwise distort the price data.

Experiments were conducted using share price data for Delta Airlines and International Business Machines (IBM) stocks traded on the New York Stock Exchange. In addition to these individual stocks, the system was tested using data from the Dow Jones Industrial Average (DJIA). Data for all stocks covers the same twenty four year period, from February 1980 to July 2004, covering just over 6000 trading days. The price values were plotted from a spreadsheet and visually inspected for anomalous values before being used as input for the GP system.

## 4.2 Experiment Aims and Objectives

The primary objective of the experimental work was to demonstrate the effectiveness (or otherwise) of this system and of the general concept - GP optimisation of TI based trading rules - in making profitable forecasts of asset price movements. In order to meet this objective it was necessary to address the following key issues:

High fitness values attained by evolved trading rules on the in-sample *training data* do not by themselves constitute proof of the predictive ability these rules. Several of the authors mentioned in the literature review apparently fail to appreciate this fact. The fitness data on its own only serves to demonstrate that the evolutionary algorithm is functioning in terms of creating individuals with increasingly high measured fitness.

It is necessary to demonstrate that the GP algorithm is learning rules which have some predictive power beyond the training period, as opposed to just learning the behaviour of the training data. With most machine learning techniques there is an issue of over-fitting learned rules to a particular set of training data. The ability of learned rules to generalise to unseen data is dependent on avoiding over-fitting, particularly when the training data is noisy. Market price data is inherently noisy - statistical tests show a high degree of randomness in stock price time-series data. According to the Efficient Market Hypothesis the movement of price values follows a random walk; therefore the data contains only noise and no information.

Positive trading returns (making a profit) also do not constitute proof of the predictive ability or practical usefulness of evolved trading rules. The experiments must demonstrate a significant (consistent) positive *excess* return relative to an appropriate performance baseline. In these experiments the baseline is the trading return resulting from the application of a naïve trading strategy over the same market period. The two naïve strategies used are buy-and-hold (for a single asset this means holding a fixed long position

for the duration of the period; the profit or loss depends solely on the difference in price at the beginning and end of the period) and the repeat rule (predict that the market will move in the same direction today as it did yesterday and trade accordingly). The comparison of trading performance relative to the two naïve trading strategies is important because:

a) Even rules which trade completely randomly will be expected to make a profit, on average in a market with a long term upward trend (if trading long positions only).
b) A high fitness evolved trading rule may be operating in exactly the same way as the simple rule, at least most of the time. It can be hard to analyse the behaviour of GP-tree trading rules; the phenomena of bloat in GP means that a very large complex tree may be functionally identical to a naive trading rule. Such a rule may be very profitable some of the time but this hardly justifies the use of GP. It does not demonstrate the usefulness of GP for this application, nor does it demonstrate the predictive power of complex TI rules.

## 4.3 Procedure

The experimental work was separated into three stages: The first set of experiments uses static evolution to optimise trading rules with fixed sequences of training and evaluation data. This stage was used to test the effect of varying the control parameters of the GP algorithm in order to optimise the performance of the system. In addition the following experimental procedures were carried out:

**Analysing the functional behaviour of GP trading rules**

In order to analyse the performance of the GP system it is useful to be able to determine how individual evolved trading rules function, in terms of how the various TI inputs affect the output BUY/SELL/HOLD recommendation. It can be hard to assess the behaviour of individual trading rules solely from examination of the structure of GP trees. Therefore the system program

includes a function which quantifies this behaviour in terms of weighting values assigned to each TI in the input set. These weights are computed by running through all possible combinations of the (ternary) input values and recording the correlation between each TI and the GP function output. In this way it is possible to test whether the optimal-fitness trading rules output by repeated training runs on the same set of data work in a consistently similar way; or alternately, to analyse how trading rules optimised for various periods of the market differ in behaviour.

**Measures to limit over-fitting**

There were two methods suggested in the literature review for limiting the tendency of the GP system to over-fit the evolved trading rules to the training data, and so improving the performance in out-of-sample trading. The first was to introduce a complexity penalising term in the fitness function (Becker and Seshadri, 2003) so that simpler rules are evolved (this could be said to fit with the principle of Occam's Razor, that simple solutions are generally better). In these experiments a different approach is taken to achieve the same aim. Instead of altering the fitness function, the system imposes an absolute limit on the allowed size (the total number of nodes) of the GP trees. This limit applies to the individuals generated in the initial population, and those produced by crossover. An alternate method was to alter the comparison function that is used in both the rank and tournament selection algorithms so that two individuals with equal fitness are ranked according to tree size: This induces an element of selection pressure toward simpler rules. Kapishnikov (2002) uses the technique of increasing transaction costs during training of the GP system in order to reduce over-fitting. This method is also tested here.

The second stage of experiments is to test the effect of periodic retraining of the GP-evolved rules on overall trading performance. The Third stage evaluates the performance of the dynamic evolution algorithm relative to the previous results for periodic retraining.

# 5 Results and Discussion

## 5.1 Static Evolution Results

Static evolution experiments were run on the IBM and Delta Airlines stock price data and the Dow Jones Industrial Average (DJIA) data. It can be seen from the plots of all three datasets that there is a qualitative change in the trends in price movements between the first 4000-4500 days and the remainder. The early period exhibits a relatively low volatility with a clear long-term upward trend while the later period shows high volatility and flat or declining trends (see figures 4, 5 & 6). It was decided to run static evolution testing separately on both periods for all three datasets, further subdividing each period into equal training and evaluation ranges, as shown on the graphs.

As there is a large relative increase in value over the first period, for all three datasets, the trading strategy for the GP-evolved rules was modified to allow the reinvestment of accumulated profits rather than executing fixed-sized trades. Also, the limit on the number of days that an in-market position can be held was removed. Otherwise it would be very hard for any technical trading rule to beat the return of the buy-and hold strategy in this period. The standard trading strategy was used in period 2 and all subsequent experiments. The strategy of continually reinvesting profits is inherently more risky, in accordance with the increased potential return, and so the results for periods 1 and 2 should not be compared directly. Furthermore, to choose a different trading strategy appropriate to each period is to make use of prior knowledge about how the market will behave - this would also invalidate such comparisons. However, the purpose of these experiments is to compare the evolved trading rules to the naïve strategies within the same trading period: Later experiments which do compare returns over the entire range of data use a fixed strategy for these reasons (a potential area of further research would be to try co-evolving trading strategies with the trading rules).

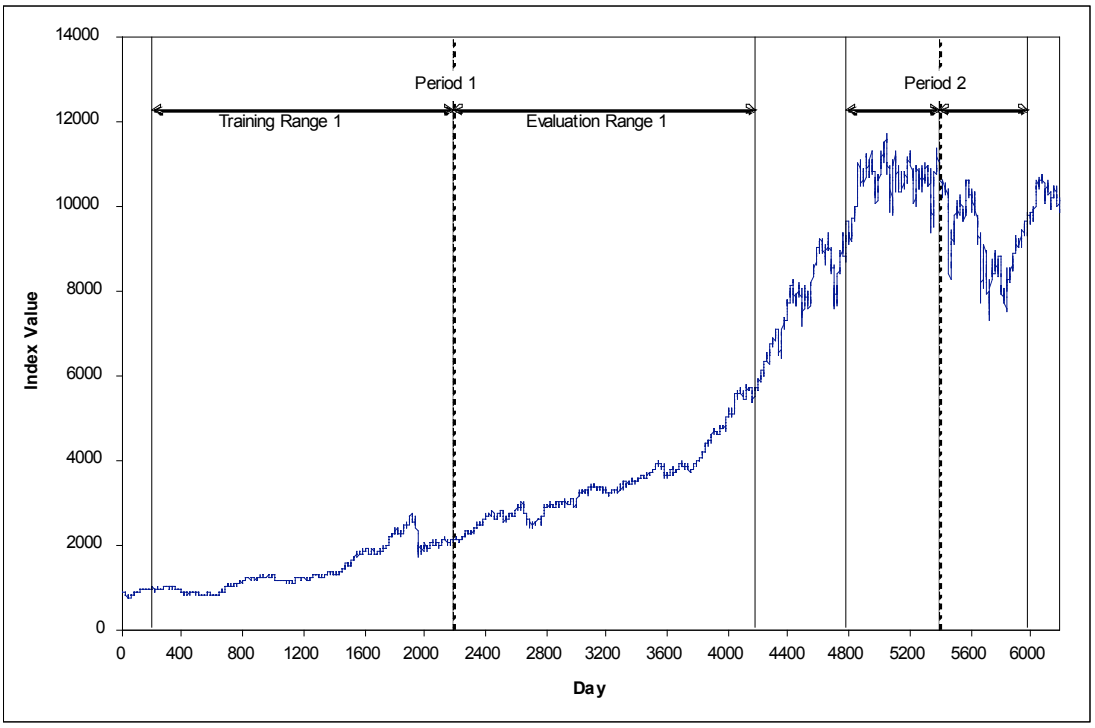


*Figure 4: Dow Jones Industrial average Feb 1980 - April 2004. The plot shows the two static-evolution testing periods with consecutive, equal length training and evaluation ranges*

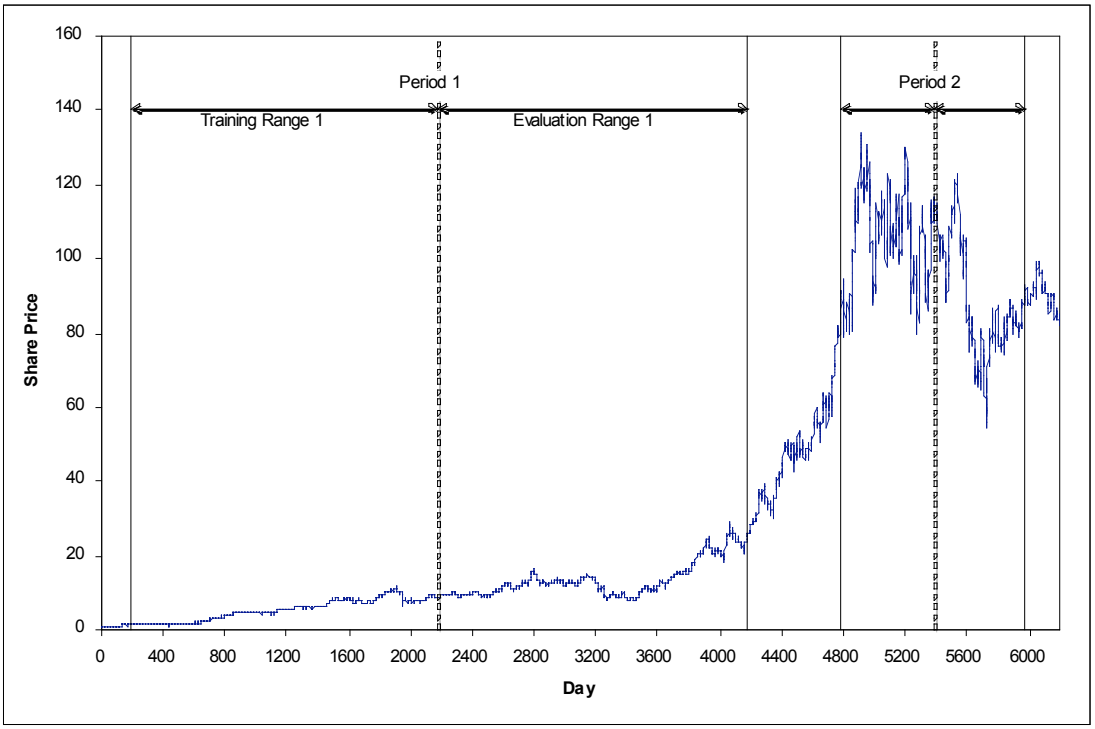


*Figure 5: International Business Machines  stock price Feb 1980 - April 2004*

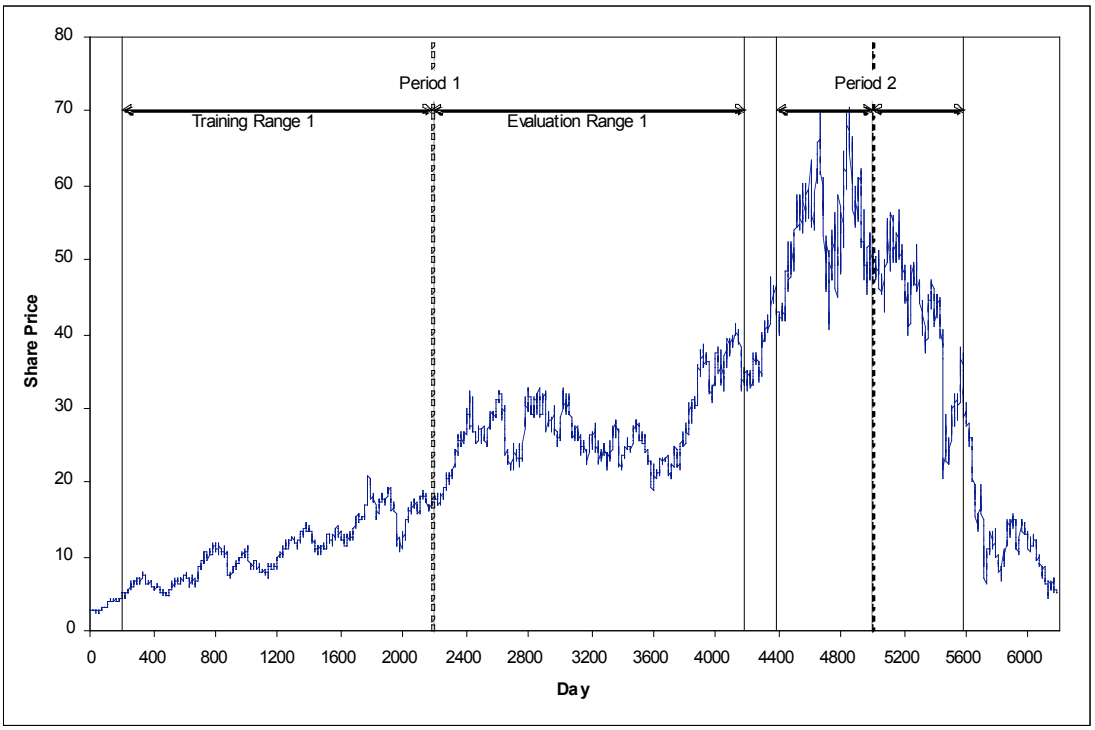


*Figure 6: Delta Airlines stock price Feb 1980 - April 2004*

### 5.1.1 Out-of-sample trading performance

Tables 2, 3 and 4 show the trading performance of the maximum-fitness evolved trading rules over a number of runs of the GP program. The results show that in period 1 the average return from the GP-evolved rules does not exceed the buy-and-hold return in *out-of-sample trading* (i.e. the evaluation range). There is a relatively large variation in these returns over the series of program runs.

In period 2 the average out-of-sample return for the evolved trading rules is positive, whereas the other strategies make a loss over the same period. However the excess average return, relative to buy-and-hold, is less that the standard deviation on this value; so this positive excess return is not statistically significant. The large variation in evaluation fitness values is what one would expect for a random trading strategy in this period, given the high volatility in price movements. Also the very high fitness values achieved in the second training period, relative to the buy-and-hold strategy and to the out-of-sample returns, indicate that the GP system is simply learning the particular sequence of price movement in the training data.

| Range (2000 day) | GP Run 1 | 2 | 3 | 4 | 5 | 6 | 7 | 8 | 9 | 10 |
|---|---|---|---|---|---|---|---|---|---|---|
| Training | 0.250 | 0.233 | 0.267 | 0.264 | 0.232 | 0.263 | 0.238 | 0.267 | 0.261 | 0.252 |
| Evaluation | 0.057 | 0.029 | 0.042 | 0.035 | 0.040 | 0.033 | 0.043 | 0.036 | 0.037 | 0.071 |

| Range (2000 day) | Average return | Standard Deviation | Buy & Hold return | Repeat Rule return |
|---|---|---|---|---|
| Training | 0.253 | 0.014 | 0.114 | 0.217 |
| Evaluation | 0.042 | 0.013 | 0.042 | 0.036 |

*Table 2: Delta Airlines period 1 trading fitness.*

| Range (600 day) | GP Run 1 | 2 | 3 | 4 | 5 | 6 | 7 | 8 | 9 | 10 |
|---|---|---|---|---|---|---|---|---|---|---|
| Training | 1.055 | 1.000 | 1.176 | 1.728 | 1.171 | 1.452 | 0.990 | 1.380 | 1.263 | 1.062 |
| Evaluation | 0.216 | 0.113 | -0.06 | 0.046 | -0.06 | -0.10 | -0.04 | 0.066 | -0.01 | -0.04 |

| Range (600 day) | Average return | Standard Deviation | Buy & Hold return | Repeat Rule return |
|---|---|---|---|---|
| Training | 1.228 | 0.235 | 0.034 | 0.102 |
| Evaluation | 0.013 | 0.097 | -0.073 | -0.293 |

*Table 3: Delta Airlines period 2 trading fitness.*

| Range | IBM | | | DJIA | | |
|---|---|---|---|---|---|---|
| | Ave. | Dev. | B & H return | Ave | Dev. | B & H return |
| Training 1 | 0.520 | 0.141 | 0.239 | 0.965 | 0.102 | 0.058 |
| Evaluation 1 | 0.083 | 0.020 | 0.096 | 0.093 | 0.025 | 0.085 |
| Training 2 | 0.381 | 0.119 | 0.052 | 1.53 | 0.262 | 0.024 |
| Evaluation 2 | -0.014 | 0.032 | -0.036 | -0.032 | 0.039 | -0.012 |

*Table 4: Fitness data for IBM and DJIA evolved trading rules.*

### 5.1.2 Effects of control parameters

The evolutionary performance of the GP algorithm was fairly insensitive to the GP control parameters: Varying the crossover and mutation probabilities, number of generations etc had little noticeable effect on the average

maximum fitness values attained for the training data. Table 5 shows the set of control parameters, the ranges tested and the values chosen for use in subsequent experiments.

| Parameter | Min | Max | Used |
|---|---|---|---|
| Crossover rate | 50 | 100 | 100 |
| Mutation rate | 0 | 100 | 25 |
| Tournament size | 3 | 10 | 3 |
| Population size | 100 | 400 | 200 |
| Generation count | 10 | 80 | 40 |
| Elite size | 0 | 20 | 8 |
| Tree size limit (node count) | 10 | 50 | 30 |

*Table 5: GP control parameter testing ranges.*

Changing the parameters of the trading system made little difference to out-of-sample average returns, except where increasing the transaction cost to the maximum of 0.5% strongly reduced these.

Testing of the various methods for limiting over-fitting as described in section 4.3 (limiting the size of GP trees, biasing selection towards smaller trees, and varying transaction costs during training) showed these to be ineffective. There was no correlation between the size of evolved GP trees and average fitness. This may be because any measure to reduce the average size of the tree structure will be more likely to affect introns, rather than functional material, and so will have little effect on the functional complexity of the trading rules.

### 5.1.3 Correlation of in-sample and out-of-sample performance

Although the trading performance of the evolved GP rules on the training data is not in itself a good measure of the ability of the system to make profitable forecasts, it is useful to measure the correlation between the in-sample and out-of-sample trading performance of individual candidate solutions.

Evolved rules did not consistently generate excess out-of-sample trading returns in the previous experiments. However, a positive correlation between

in-sample and out-of sample returns would suggest that this may be achievable by some improvement of the GP system: In this case it would be worthwhile investigating ways to achieve higher training fitness, as this should lead to greater excess returns on out-of-sample trading, either by optimising the parameters of the evolutionary algorithm (although this has already been tried), or by using a different TI input set. Conversely, a negative correlation would indicate over-fitting and measures could be taken to reduce this effect.

Experimental results show no significant correlation between in-sample and out of sample trading profits. Figures 7 and 8 show the training and evaluation fitness values for the trading rules output from a number of GP runs (the rules having the maximum attained training fitness from the evolutionary population).

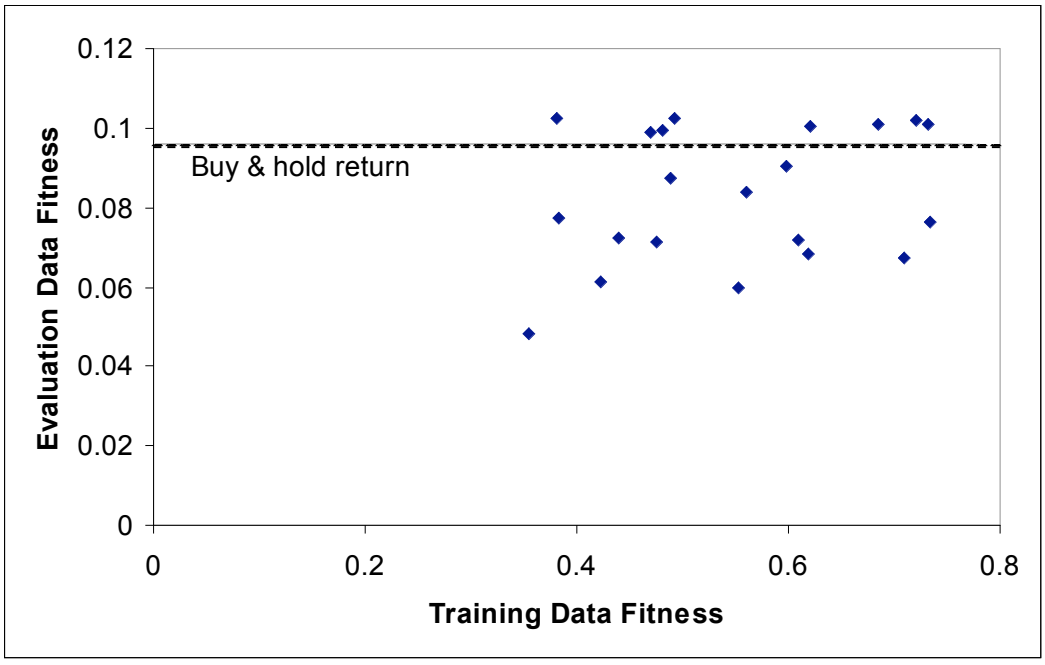


*Figure 7: Correlation of training fitness to out-of sample trading fitness for evolved GP rules. (IBM data; period 1)*

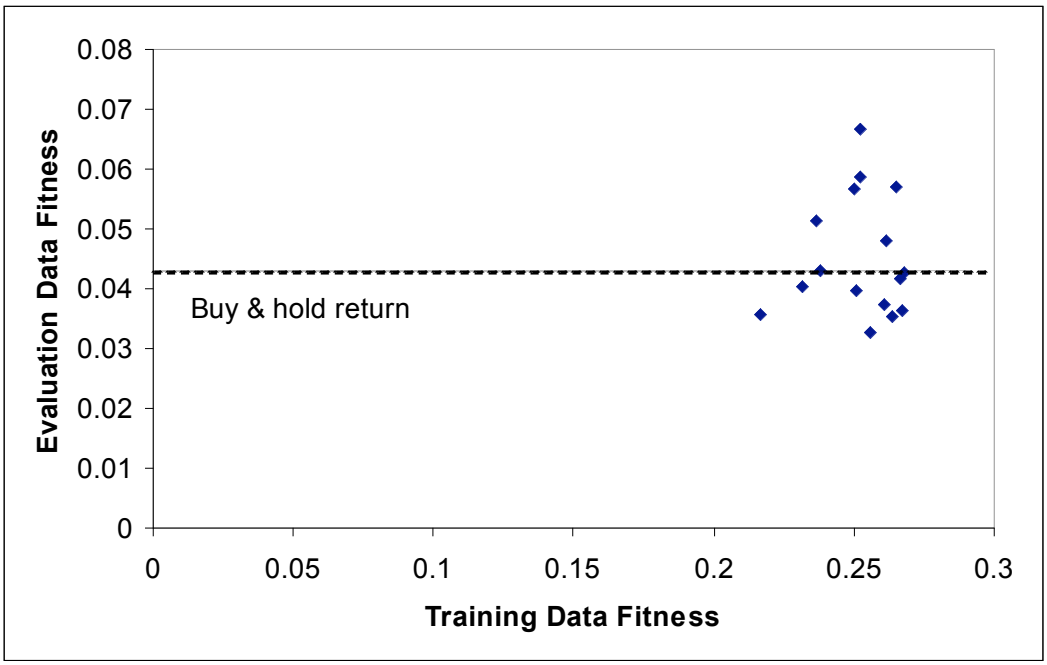


*Figure 8: Correlation of training fitness to out-of sample trading fitness for evolved GP rules. (Delta Airlines data; period 1)*

### 5.1.4 Weighting of TI in evolved trading rules

Figures 9 and 10 show the relative weightings of the various TI in the maximum-fitness evolved trading rules, over a series of program runs, for the DJIA and Delta Airlines data respectively.

For the DJIA trading rules, generated from repeated static evolution runs on the same set of training data, the weighting given to the various TI appears to be quite random. One would expect certain TI to perform better than others on certain ranges of data, and these would receive consistently higher weighting in the optimised trading rules. This does not seem to be the case in the trading rules evolved from period 2 of the DJIA data (figure 9).

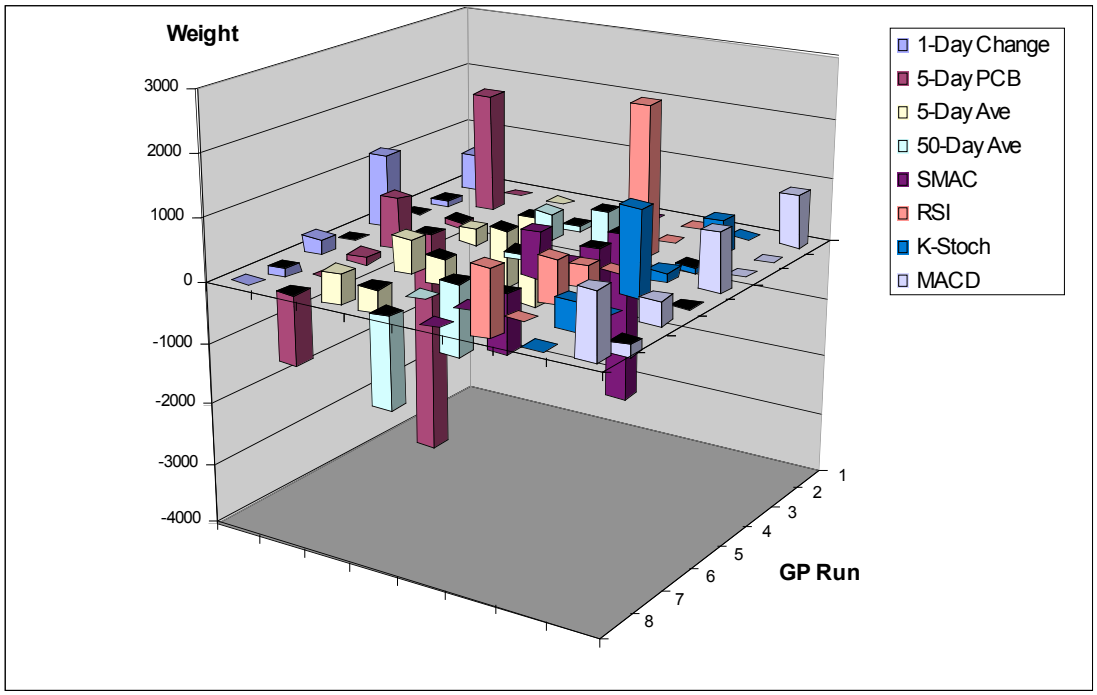


*Figure 9: Relative weighting of TI in fittest trading rules over a series of evolutionary runs. (DJIA data; period 2)*

The results from section 5.1.1 show that for Delta Airlines the repeat rule trading strategy outperforms buy-and-hold in both of the training periods. This is a noteworthy result in itself: such a strategy would have to be very effective in predicting the direction of daily price movements as it trades frequently and thus incurs high transaction costs. Figure 10 shows that GP-evolved trading rules trained on the same data consistently give a high weighting to the 1-day price change indicator. This means that these rules are acting according to the naïve repeat rule strategy - “predict that the price will move in the same direction tomorrow as it did today” - most of the time.

This result demonstrates that GP system is able to learn to use individual TI where they are known to perform well, and does so repeatedly. However, the random weightings of other TI for both sets of results indicate that they do not have any consistent predictive ability. Another point to note is that figure 9 shows that the TI are just as likely to have negative weightings as positive (meaning that there is a negative correlation between the TI value and the output of the trading rule). This seems to contradict the idea that the TI rules

are good indicators of what price trends are going to do: If the PBC rule described in section 2.1.1 is really correct how come the GP system evolves rules which predict the market will go in the opposite direction to the PCB indicator?

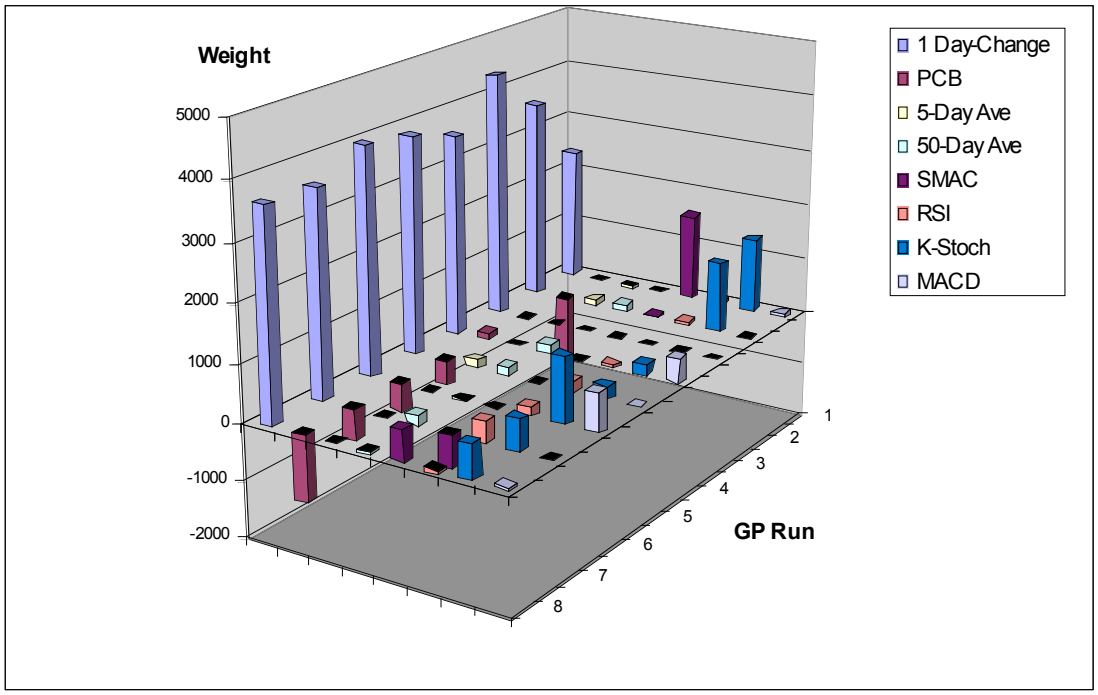


*Figure 10: Relative weighting of TI. (Delta Airlines data; period 2)*

## 5.2 Static Evolution with Periodic Retraining

In these experiments simulated trading was run over a range of data covering both periods 1 and 2 from previously. This range was divided into a set of consecutive equal-length periods. GP training was run on each period separately, producing a sequence of optimised trading rules - one for each period. The fitness values obtained for each period resulted from the application of the rules trained on the one immediately preceding it.

### 5.2.1 Periodic Retraining Results

Figures 11, 12 and 13 show the performance of GP-evolved technical trading rules relative to buy-and-hold and repeat rule strategies over consecutive 400 day periods for Delta Airlines, IBM and the DJIA respectively. The fitness

values for the evolved rules for each period are averaged over a number of training runs, as in section 5.1.1. The figures also show the returns from the strategies averaged over all periods (day 601 to 5800), excluding the first period for which there is no GP-evolved rule for comparison. It should be noted that the average buy-and-hold return calculated in this way is not the same as the return which would result from continuously holding the asset over the entire time range, due to the effect of compounded capital growth.

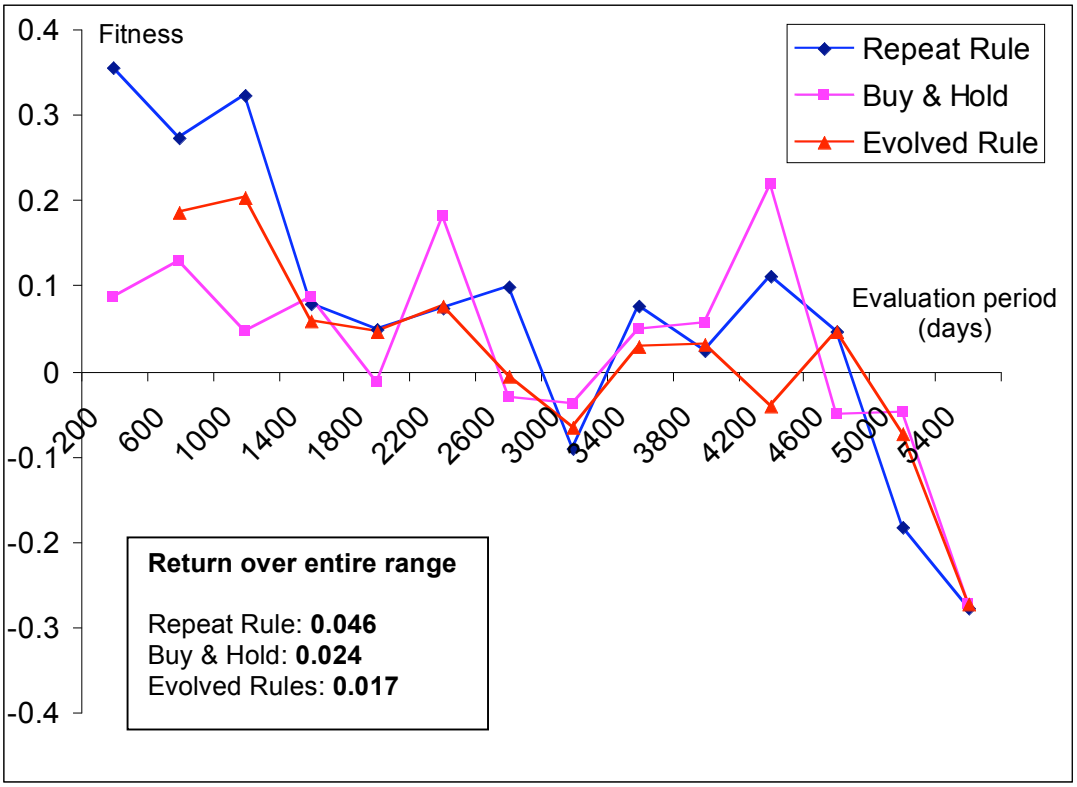


*Figure 11: Relative performance of evolved trading rules to buy & hold and repeat strategies trading Delta Airlines stock with 400 day consecutive training and evaluation periods.*

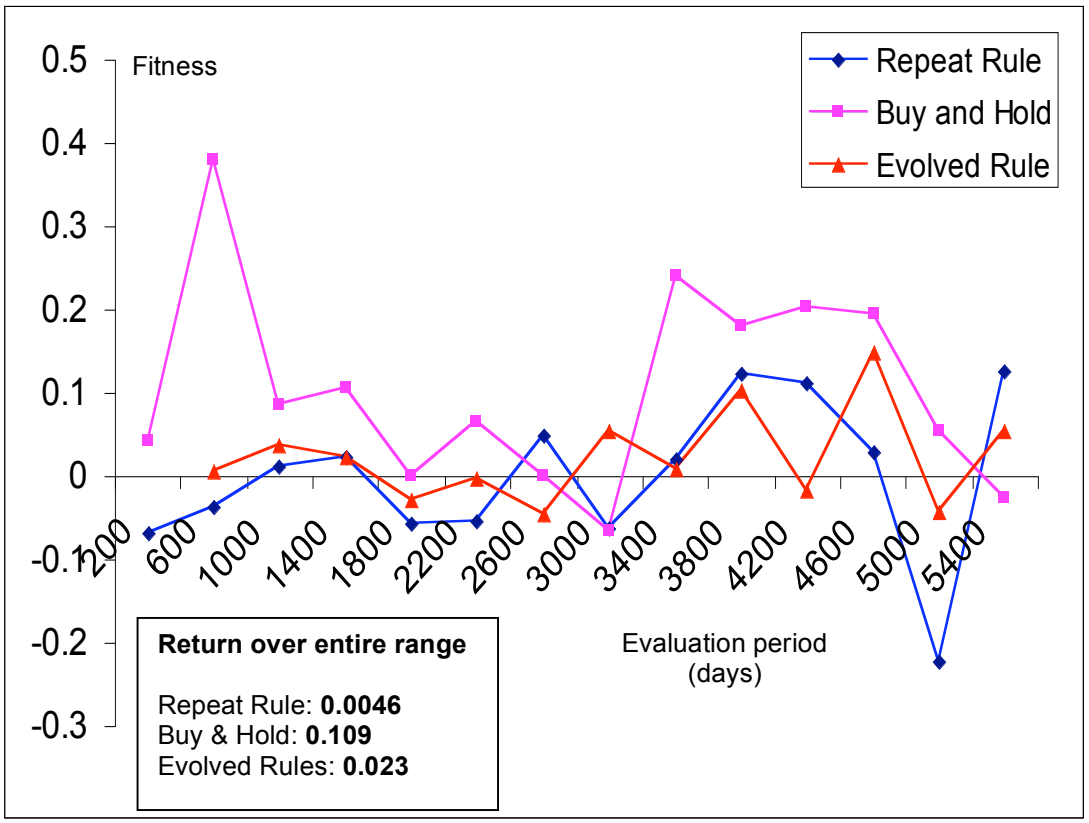


*Figure 12: Relative performance of evolved trading rules to buy & hold and repeat strategies trading IBM stock with 400 day consecutive training and evaluation periods.*

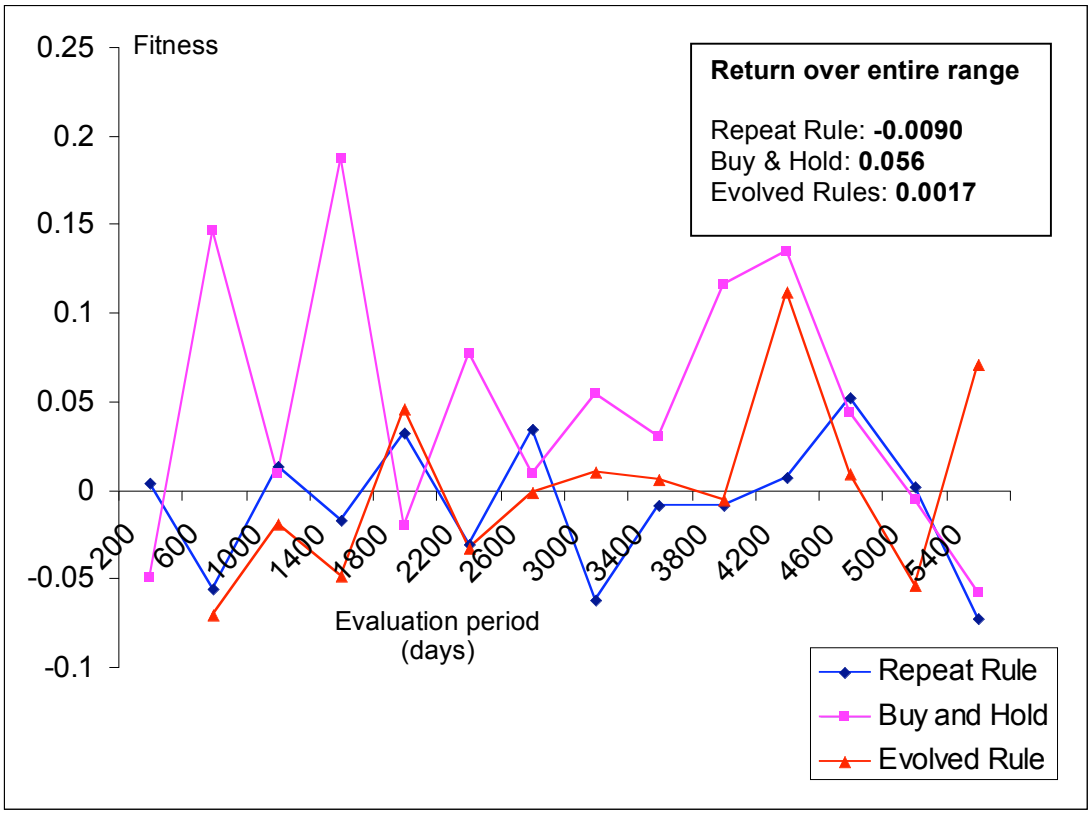


*Figure 13: Relative performance of evolved trading rules to buy & hold and repeat strategies trading the DJIA with 400 day consecutive training and evaluation periods.*

It can be seen that the sequence of evolved trading rules produce negative excess returns in comparison to buy-and-hold for all three datasets when taken over the entire trading range. The results show that, as noted before, the repeat rule strategy only performs well in trading with the Delta Airlines stock.

There are individual periods when the evolved rules are seen to outperform buy-and-hold. However, as mentioned, the single-period return values for the GP-evolved rules are averages only, and the buy-and-hold values are generally well within the measured standard deviation of these averages. The one occasion where the evolved rules produce statistically significant positive excess returns is in the latest trading period for the DJIA (see figure 13). A possible explanation for this result comes from the fact that, as can be seen in figure 4, the DJIA index data for this period (days 5400-5800) exhibit high

volatility. There are two particularly sharp drops in the index value that occur in this range. Examination of the TI weighing in the technical trading rules used in this period shows a consistently high weighting for the 50-day moving average indicator. Application of this individual TI rule would cause the system to sell the index prior to these drops and this alone would account for the excess returns observed in this period. In this one occasion a single TI which is found to be effective in one training period also produces profits in the following evaluation period. In itself this result hardly constitutes a validation of Technical Analysis techniques.

Figure 14 shows the results from periodic retraining on the Delta Airlines data using a 600 day training/evaluation period. Extending the amount of training data in this way may be expected to increase the ability of the GP system to evolve profitable trading rules at the expense being able to adapt quickly to changing market conditions. Either way, the evolved rules still do not outperform either naïve strategy.

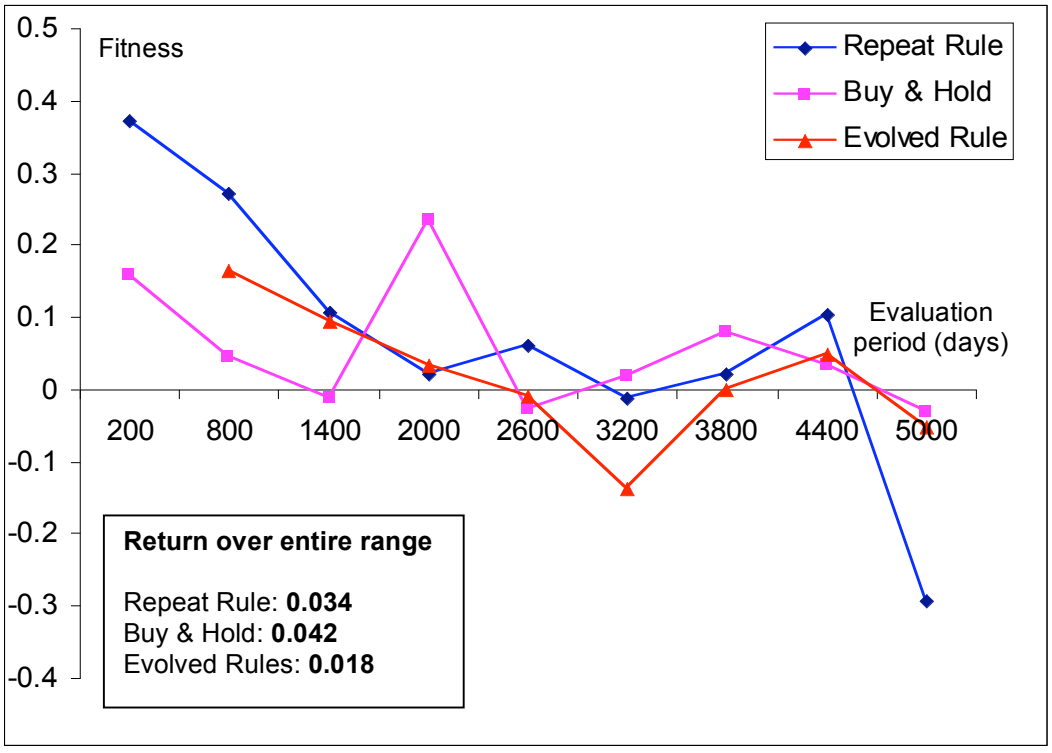


*Figure 14: Trading results for Delta Airlines with 600 day training and evaluation periods*

### 5.2.2 Measuring time variation in the performance of static evolved rules

Figure 15 shows the period-by-period fitness values of static-evolved trading rules - trained on each of the first three periods - over (almost) the full range of data for the DJIA. The purpose of this test was to see how the performance of individual trading rules changes over time. This was supposed to reveal the way in which market conditions evolve - whether they change incrementally or in discrete steps. The fitness data does not reveal any useful information in this regard. Fitness values show neither a decreasing trend nor a sudden change in any period but change randomly over time. The raw price data (figures 4, 5 & 6) appear to show a discrete change in behaviour at around 4800 days, as mentioned before. This change is not reflected in the fitness data in figure 15. It can be seen that the trading rule evolved in the third period (1001-1400 days) makes a loss in the next period (1401-1800 days) and is most profitable over 4600-5000 days.

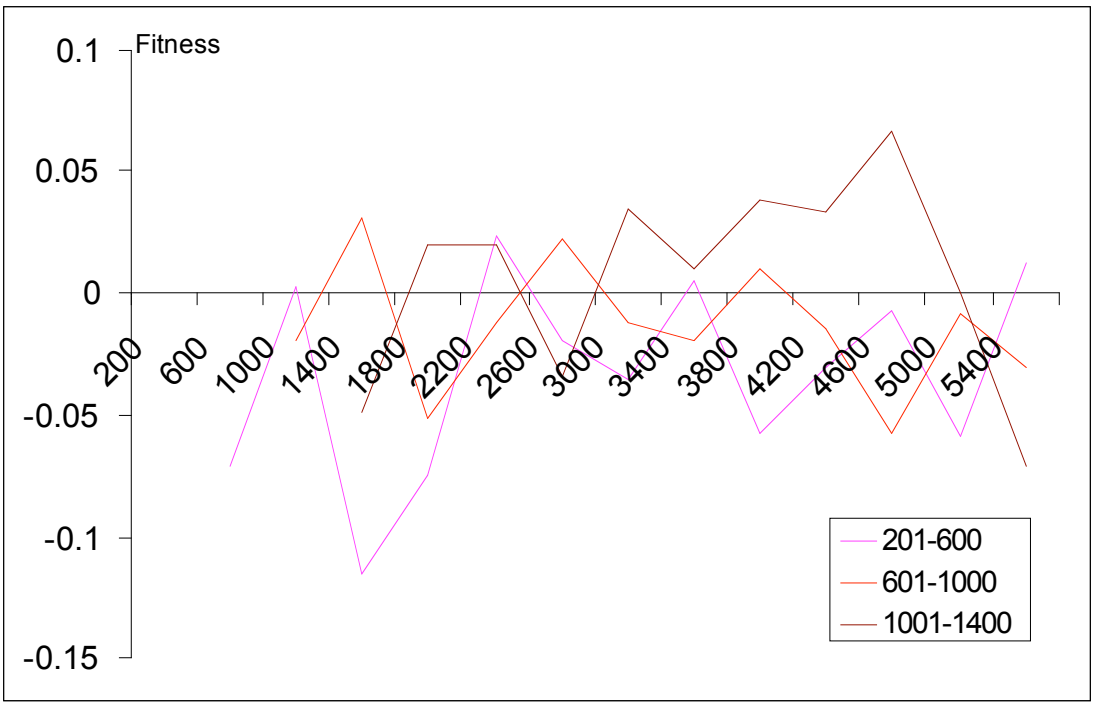


*Figure 15: Out-of sample fitness for trading rules evolved in the first three 400-day periods (DJIA data).*

## 5.3 Dynamic Evolution Results

The static evolution algorithm was run over the same range of data as used for the periodic retraining experiments (day 601 - 5800). Table 6 shows the period-by-period trading fitness for the system using the dynamic algorithm. Average fitness values are calculated for each period from a number of runs of the program, along with the corresponding standard deviation. The fitness values for the repeat rule and buy-and-hold strategies are show for comparison, along with the results from the previous experiments using static evolution with periodic retraining.

It can be seen that the system does not outperform either naïve strategy on average over the entire time range. This is not surprising, given the results from the previous experiments which all indicate that GP evolved technical trading rules are unable to generate excess profitable returns. The dynamic algorithm did produce better returns than periodic retraining, over the whole trading range. There is also a smaller variance in the returns for dynamic evolution, hence a superior risk adjusted return.

| | | | Dynamic Profit | | Stat evol rule profit | |
|---|---|---|---|---|---|---|
| **Range start day** | **repeat rule** | **buy and hold** | **Ave.** | **Std. Dev.** | **Ave.** | **Std. Dev** |
| 600 | 0.272 | 0.128 | 0.266 | 0.053 | 0.186 | 0.072 |
| 1000 | 0.322 | 0.046 | 0.105 | 0.049 | 0.204 | 0.045 |
| 1400 | 0.079 | 0.087 | 0.110 | 0.032 | 0.059 | 0.061 |
| 1800 | 0.048 | -0.014 | -0.027 | 0.040 | 0.045 | 0.063 |
| 2200 | 0.073 | 0.181 | 0.093 | 0.022 | 0.075 | 0.028 |
| 2600 | 0.098 | -0.030 | 0.088 | 0.039 | -0.007 | 0.022 |
| 3000 | -0.092 | -0.039 | -0.111 | 0.050 | -0.065 | 0.107 |
| 3400 | 0.075 | 0.049 | 0.114 | 0.078 | 0.028 | 0.072 |
| 3800 | 0.023 | 0.056 | 0.024 | 0.021 | 0.032 | 0.049 |
| 4200 | 0.112 | 0.217 | -0.061 | 0.084 | -0.040 | 0.054 |
| 4600 | 0.045 | -0.050 | 0.024 | 0.026 | 0.046 | 0.085 |
| 5000 | -0.182 | -0.049 | -0.035 | 0.029 | -0.073 | 0.052 |
| 5400 | -0.279 | -0.272 | -0.286 | 0.055 | -0.273 | 0.116 |
| **Average** | **0.046** | **0.024** | **0.023** | **0.044** | **0.017** | **0.064** |

*Table 6: Comparison of dynamic evolution algorithm trading returns to naïve rules and static evolution with periodic retraining (Delta Airlines).*

Figure 16 shows the average and maximum fitness values for the dynamically evolving population. These fitness values are retrospective, in the sense that they are evaluated at each time step from the preceding range of historical data, so they represent the ability of the system to adapt to the latest available information. They do not give any direct indication of the forward predictive ability of the system - this is shown in table 6. It appears, from comparing these values to the raw market data in figure 6, that the fluctuations in fitness are closely correlated to the direction of the price trend. Wherever there is a particularly sharp drop in the stock price there is a corresponding decrease in the average fitness value. The data shows that on several occasions where these drops have occurred, at 1700, 4000 and 4800 days, there is an *increase* in the maximum fitness. The effect is not very clear but it is repeated in further runs of the program.This may be an indication that the algorithm is able to rapidly adapt a small proportion of the individual candidate solutions to the latest available data. This is exactly what is required, as the trading recommendations of the system as a whole are based on the current fittest member. However there are other occasions when the two fitness values move together. The effect of large Individual movements in asset price appears to dictate changes in fitness rather than any supposed change in underlying market conditions.

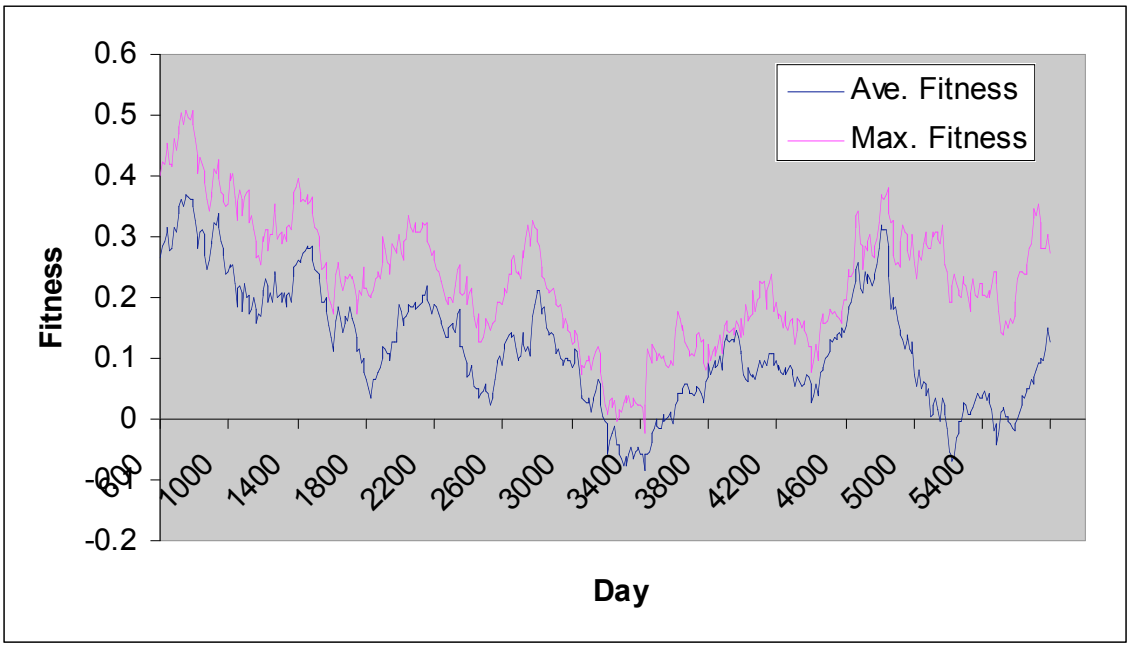


*Figure 16: Fitness statistics for the dynamic GP algorithm (Delta Airlines input data).*

# 6 Conclusions

My work clearly demonstrates that GP-evolved trading rules are not effective in predicting future movements in asset price when using Technical Indicator information derived solely from historical price data. This conclusion should be qualified by saying that the investigation only covered the use of daily trading data for individual shares and for one market index; it is possible that the technique may be more successful when applied to other classes of assets or markets, or if higher frequency 'tick' data is used. There has been other work done in these areas and, as mentioned in the literature review, the results appeared to be mixed, at best.

It is also possible that the system was simply using the wrong set of Technical Indicators. Looking on Yahoo Finance and other finance-related websites reveals a huge range of TI together with information on how and why they are supposed to work. My choice of TI was primarily based on those used in the trading systems of other researchers as mentioned in the literature review. In fact, in emulating the work of several authors, a wider range of TI was used than those others did individually. The experiments also investigated the use various subsets of the chosen TI in runs of the GP algorithm - as shown, this had little effect on trading results.

This approach to market forecasting may be more successful if different sources of information were used as input for the GP learning algorithm. These could include data from different markets, for example bond rates, commodity prices or FX rates, in order to predict stock market trends or vice versa. Potentially any other source of information which is judged to be relevant - such as interest rates or data from fundamental analysis - could be incorporated. The GP program developed here could easily be adapted to take input from different sources. There has been work done in the use of GA and GP to generate trading rules based on market indicators other than pure Technical Analysis. Bauer (1994) used GA to develop strategies based on

bond rates. Yet again, there was no conclusive demonstration of significant excess profits on out-of-sample data.

Apart from the obvious application of trying to make money, investigating the use of GP with these different sources of learning input would be a useful tool for discovering relationships between asset prices and other economic factors. Such information may be important in itself for the study of economics. More generally, GP can be a useful tool for discovering complex relationships between wide classes of variables if approached from the point of view of asking: “Can a GP learn to predict variable X, with a significant probability, using input data from variable Y”.

In these terms, the results shown here appear to demonstrate that Technical Analysis is simply not effective as a methodology for forecasting market trends. This implies that no useful information is contained in historical price data. As stated in section 2.1, evidence that technical trading rules can be profitable is evidence against the Efficient Market Hypothesis; the converse is also true. Therefore my results appear to support the EMH.

Regarding the performance of the dynamically adaptive GP algorithm; although this was no more successful in predicting the future movement of market prices than the method of periodic retraining, the output fitness data does indicate that it was adapting the population of candidate solutions over time according the variations in fitness statistics. The algorithm appeared to be able to adapt to changes in market data in a retrospective sense, although evidence for this was inconclusive. The fact that the trading performance was poor is not surprising considering that the preceding experiments already demonstrated that TI had no useful forecasting ability for the data used. This being the case, the best that could be hoped for is that the algorithm could maintain a high level of fitness on the latest available historical data. The results from the experiments with the dynamic GP algorithm suggest that the basic concept is sound, so it may be applied successfully to other problem domains and this could be investigated in future work.