Comprehensibility, Overfitting and Co-Evolution in Genetic Programming for Technical Trading Rules

by

Mukund Seshadri

A Thesis

Submitted to the Faculty

of the

WORCESTER POLYTECHNIC INSTITUTE

in partial fulfillment of the requirements for the

Degree of Master of Science

in

Computer Science

May 2003

APPROVED:

_____

Dr. Lee Becker, Advisor

_____

Dr. Carolina Ruiz, Reader

_____

Dr. Micha Hofri, Head of Department

Abstract

This thesis presents Genetic Programming methodologies to find successful and understandable technical trading rules for financial markets. The methods when applied to the S&P500 consistently beat the buy-and-hold strategy over a 12-year period, even when considering transaction costs. Some of the methods described discover rules that beat the S&P500 with 99% significance.

The work describes the use of a complexity-penalizing factor to avoid overfitting and improve comprehensibility of the rules produced by GPs. The effect of this factor on the returns for this domain area is studied and the results indicated that it increased the predictive ability of the rules. A restricted set of operators and domain knowledge were used to improve comprehensibility. In particular, arithmetic operators were eliminated and a number of technical indicators in addition to the widely used moving averages, such as trend lines and local maxima and minima were added. A new evaluation function that tests for consistency of returns in addition to total returns is introduced. Different cooperative coevolutionary genetic programming strategies for improving returns are studied and the results analyzed. We find that paired collaborator coevolution has the best results.

Acknowledgements

If I have seen further [than certain other men] it is by standing upon the shoulders of giants - Isaac Newton (The Columbia World of Quotations. Copyright 1996 Columbia University Press.)

None of this work would have been possible without the unfailing enthusiasm, patience and tolerance of Prof. Lee Becker. His contributions and support are greatly valued and appreciated and I am deeply indebted. I would also like to thank Prof. David Brown and Prof. Carolina Ruiz for their feedback and Prof. Rundensteiner and Prof. Hofri for their support throughout my course of study.

My father introduced me to technical analysis and the desire to ask questions and my mother strongly encouraged my graduate studies. My brother has helped me be competitive and they deserve a lot of credit. Their help and good wishes are deeply appreciated and acknowledged. This work is dedicated to them.

Thanks are also due to Jignesh Patel, who was a constant companion in my long sojourns through Computer Science and Niloufer Rodrigues who helped me in various ways and by providing copious amounts of tea and coffee.

-Mukund Seshadri

Table of Contents

List of Tables and Figures

List of Tables

List of Figures

1 Introduction

The last few years have seen the increasing application of Genetic Programming to computational finance. A lot of this work has been in the area of financial trading and prediction. This has been especially true in the area of stock price and foreign exchange rate prediction. Attempts have been made at mining financial time series data for patterns that may be applied to the future. The development of Genetic Programming over the last decade, its inherent parallelism and its ability to quickly find near-perfect solutions quickly for NP-hard problems, has encouraged the application of its techniques to this area. In particular, technical analysis is such an area that offers a set of building blocks for pattern detection and is utilized by investors in attempts to forecast the market and codify trading decisions.

Most studies however have suffered from being either unable to beat the buy-and-hold strategy and more seriously the lack of simple, understandable technical trading rule patterns. The inherent randomness of the Genetic Programming procedure also creates problems in the variability of returns and methods that return consistent performance are sought.

This study attempts to solve these problems through different choices made for the Genetic Program and apply the techniques of co-evolution to further improve on work already done.

The remainder of the thesis is organized as follows. Chapter 2 introduces Genetic Programming and Technical Analysis and details related work in those areas. Chapter 3 lists the improvements suggested over previous work and describes experiments to test them. Chapter 4 tabulates the results of the study and Chapter 5 concludes with observations on the results.

2 Background

This chapter gives a basic overview to Genetic Programming and the problem domain (Technical Analysis). It lists various technical indicators that are used by technical analysts and describes their usage. Previous studies on technical analysis are studied and work on generating technical trading rules by Genetic Programming is discussed. The problems with these previous approaches that we will attempt to address are identified and various solutions to address them are discussed.

2.1 Genetic Programming

Evolutionary algorithms are computer-based problem solving systems that are based on biological evolution as key elements of their design and implementation. They can be further classified into Genetic Algorithms, Evolutionary Programming, Evolution Strategies, Classifier Systems and Genetic Programming. Of these Genetic Algorithms (GAs) and Genetic Programming (GP) are closely related.

2.1.1 Introduction to Genetic algorithms

GAs were invented by Holland [1] in the 1960s as a method to study evolution. GAs mimic the biological evolution of organisms by following the "survival of the fittest" paradigm. Initial solutions are generated randomly and their fitness tested. This fitness is evaluated by a fitness measure called an objective function that numerically indicates how well a solution satisfies the problem. The fittest solutions are chosen

for mating in proportion to their fitness. In the reproduction process, the components of the solution are swapped and those children are also evaluated. A small percentage of solutions are also subjected to mutation to maintain diversity in the gene pool. Then the fittest among all these are chosen and form the next generation where the steps are repeated. At the end of a number of generations, the fittest organism is chosen as the answer. This sort of population-based algorithm with crossovers and mutations was a major innovation.

As a simple example of genetic algorithm consider the problem of finding the positive roots of an equation [x^3 + 3x^2 + 3x +1]. The GA designer might decide to represent the solution in the form of an 8-bit array. Therefore the representation of the solution to the problem would be

0000 0001 1

where the number to the left is the binary representation of the number to the right. Initially the GA would start out with a population of random bit strings. For example

1010 1000 168

0000 1111 15

…

1111 0101 245

Then it would measure the fitness of each of these organisms. In this case a valid fitness measure would be to minimize the distance between 0 and the value of the

expression for the particular organism/genome. Therefore the fitness evaluations would be

1010 1000 -168

0000 1111 -15

…

1111 0101 -245

The next step would be to choose the fittest individuals and mate them. This would mean crossing over 168 and 15, by choosing a random point for crossover. For example if we chose 4 left bits and 4 right bits we would have

1010 1000 x 0000 1111 = {1010 1111, 0000 1000}

as can be seen this generates 0000 1000 which is closer to the solution than 245 and will replace it. In this manner proceeding along, generation after generation we will hopefully get to the right answer. The GA algorithm is depicted in Fig.1 and outlined in the algorithm below

1 Initialize a population of randomly created individuals

2 Until an individual is evolved whose fitness meets some pre-established criterion:

- **2.1** Assign each individual in the population a fitness, based on some domain-specific fitness function.
- **2.2** Set the "child population" to the empty set.
- **2.3** Until the size of the child population equals that of the parent population:
  - **2.3.1** Select two members of the parent population, with the probability of a member being selected being proportionate to its fitness (the same member may be selected twice).

2.3.2 Breed these two members using a crossover operation to produce a child.

2.3.3 (Possibly) mutate the child, according to some pre-specified probability.

2.3.4 Add the new child to the child population.

2.4 Replace the parent population with the child population.

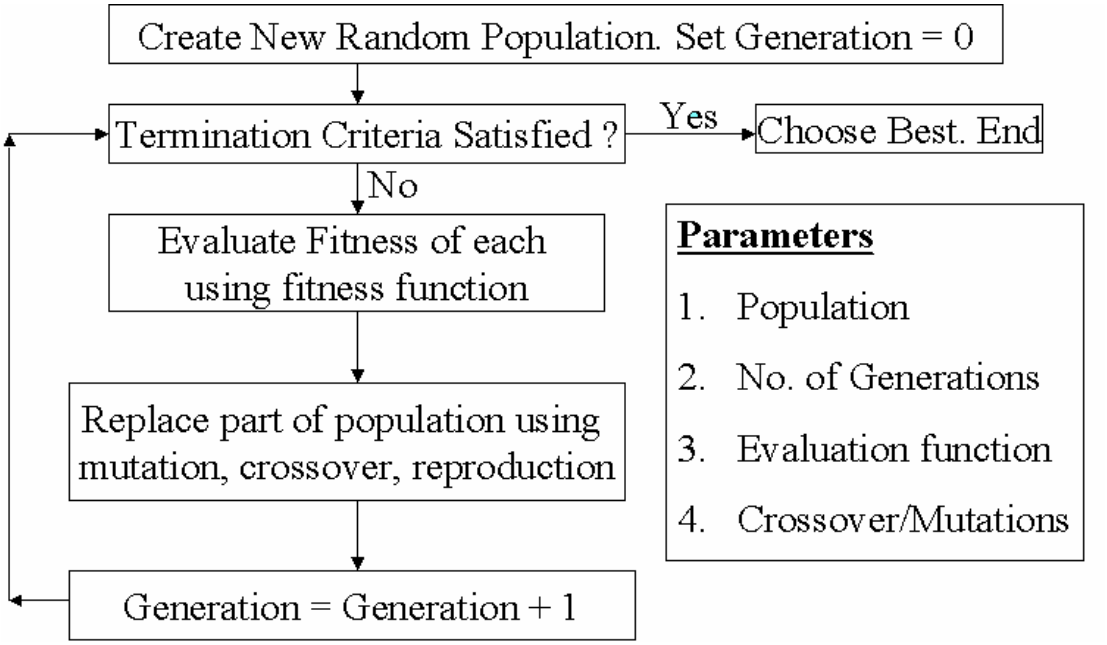


Figure 1: Flowchart for GA algorithm

Because the best parts of the solution are carried over from generation to generation, the near optimal solutions tend to be found. Because the solutions are evaluated in parallel, GAs are used for NP-hard problems where they can find approximations to the best solution pretty quickly. The theoretical foundations of this process lie in The

Schema Theorem [1]. A schema is a set of chromosomes that share certain values. The schema theorem relates the fitness of members of a schema to the expected number of schema members in the next generation. Members of schemas that have successful characteristics tend to increase in number. GAs have been successfully used in different areas including schedule optimization [2].

2.1.2 Introduction to Genetic Programming

GP is different from genetic algorithms in that the forms of the organisms/genes are trees. The idea was first introduced by John Koza [3] to evolve LISP programs where the tree was the program syntax tree. For example a genome for a genetic program to evolve an expression might look like Fig. 2.

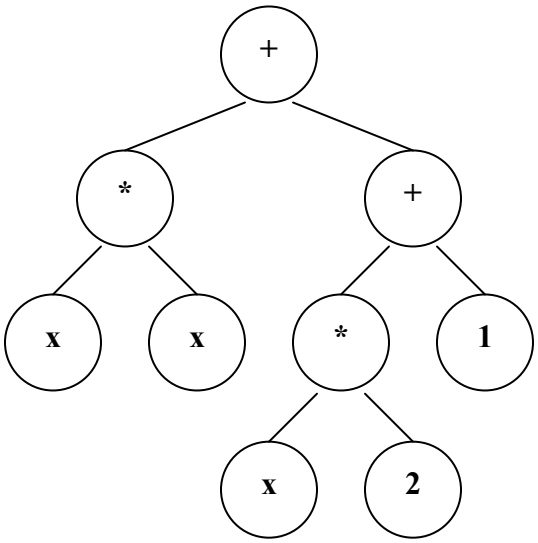


Figure 2: Example of a GP genome

This particular tree represents the expression x^2+2x+1. As a genome it can be used for curve fitting where the evaluation function might be the inverse of the sum of absolute differences between the tree value and the actual value. For crossover

operations one could replace a random sub tree of one tree with another as illustrated in Fig.3. For mutation one could destroy the sub tree, or change a node to another node. This is a very powerful structure that has been used in a wide number of areas. They have been used to classify one-dimensional two-state cellular automata [4], in which they have performed better than all human generated rules and other automated rule generation approaches in the classifying of one-dimensional two-state cellular automata and also for near-minimum-time control of a spacecraft's altitude maneuvers [5]. In addition, of the nodes of the tree are operators and the data is stored in the nodes the tree can represent a rule. These have been used in reinforcement learning as in [6] to develop rules of the solution by reinforcement.

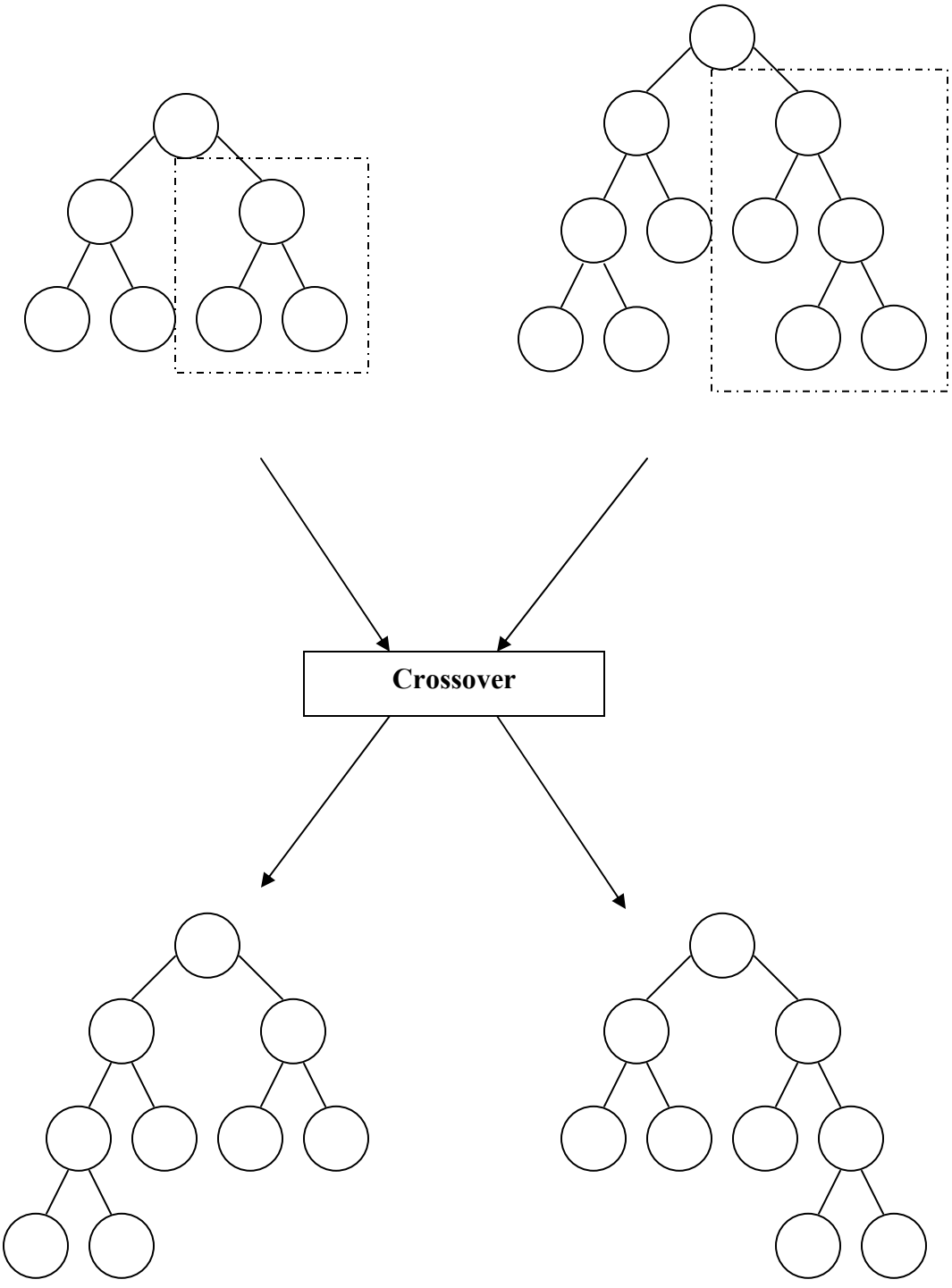


Figure 3: Crossover for GP trees

2.2 Coevolution

The concept of biological systems has been further extended as in [7] to simultaneously co-evolve two different species that depend on each other to achieve a solution. According to geneticists Futuyuma & Slatkin [8] following Jansen [9], "a rigorous definition of coevolution requires that a trait in one species has evolved in response to a trait of another species, which trait was itself evolved in response to the first species." This more closely represents natural evolution where species evolve characteristics in response to other organisms and not just independently of them. In a way it is equivalent to having a constantly changing environment instead of a stable one where the environment consists of the same factors as in single evolution but now includes other species. The most familiar example of co-evolution is parasitism and symbiosis. Another example is the evolution of deer and lions where both need to learn to run faster. The lions need to run faster to catch their prey and the deer need to get away from them. The slowest lions fail to catch their prey and will stand a greater chance of going hungry. This would mean that the faster lions have a better chance of reproducing and passing on their genes. Similarly the slowest deer would be preyed on thus reducing their ability to pass on their genes. The faster ones on the other hand bear children and the average ability to run faster of the overall population of deer improves.

This is an example of competitive coevolution. When applied to genetic programming it would start out with two or more populations, each randomly created. Then the algorithm proceeds as detailed above in genetic algorithms but with the change that

the evaluation function of each species consists of testing the performance of each individual with each individual (or a sampling) from the other populations. No organism has therefore an absolute fitness measure but only one that depends on the organisms it is evaluated against. This ensures an "arms race" [Angeline and Pollack10] between the populations with each one getting better and better. These are known as "coupled fitness landscapes " [11] because the fitness curve of each population is affected by the organisms in the other population. There have been very interesting applications of this idea as in Hillis' [12] sorters vs. sequences where the sorting networks evolve as one population and the other population consists of sorting problems which evolve to get more difficult.

Coevolution introduces the credit assignment problem. In competitive coevolution credit is easy to assign. The score for one organism is the complement for the organism it is evaluated with. For cooperative co-evolution however, because many parts make up the solution, and they have to be evolved separately, thought must be given to how the credit for a successful solution is passed down to its components. For example, Holland [13] describes a bucket brigade system for a classifier system in which the co-evolving organisms are classifier rules.

Coevolution also forces one to consider the collaborators to use for evaluation. Potter & De Jong[14] describe an architecture for cooperative coevolution, in which individuals in one species are evaluated by using collaborators from each of the other coevolving species. There exist a number of possibilities for choosing collaborators.

All individuals of each of the other species [15] can be used for evaluations but this is time-consuming and can lead to combinatorial explosion if there are many species. To reduce the number of evaluations and pairings considered, it might be possible to use the best individuals from each of the other species as collaborators. This has been tried by [16]. In some cases randomly drawn individuals of each of the other species in the same generation as in [17] and [18] have been experimented with. [13] reports work done on choosing fixed partners from the other species of the same generation.

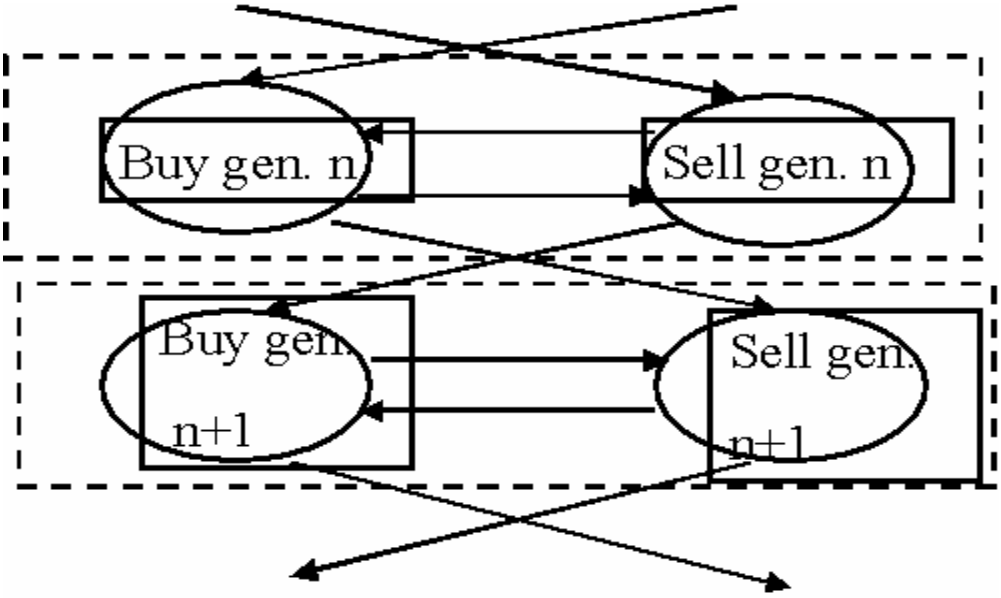


Figure 4: Possible choices of collaborators

Fig.4 above succinctly outlines the various combinations that can be exploited in choosing collaborators.

2.3 Comprehensibility and Overfitting

Trees evolved by GP can be very large. For example, Allen and Karjalainen[19] find rules that vary from 9 to 94 nodes and with a minimum depth of 5. Consider the tree they describe in their paper (Fig. 5). This tree represents the rule

*price * minimum (price) – 0.0688 * price > 0.893*

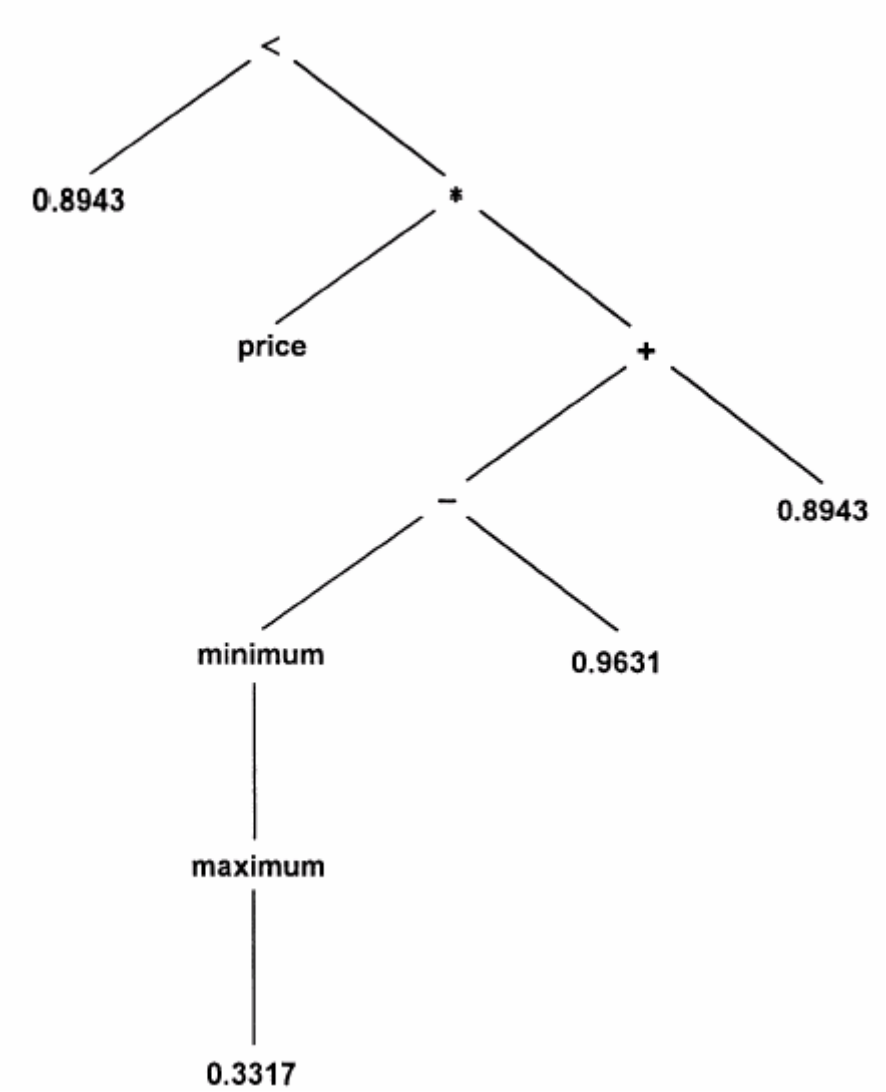


Figure 5: Technical trading rule from Allen and Karjalainen's paper

There are a few problems with trying to understand this rule.

1. Non-intuitive nature of the rule: The meaning of the rule is not immediately obvious by looking at the tree. On noting that this is one of the smaller trees the problem becomes more serious.
2. Redundancy: The papers discusses the structure of the trees and notes that there are a lot of redundant sub trees in the rules and even sub trees that are never visited during evaluation.
3. The size of the tree. This is one of the smaller trees. As the trees get larger, they become more difficult to understand and they become more prone to over fit training data.

These problems are discussed in more detail below.

2.3.1 Comprehensibility

Consider the structure of a technical trading rule that is to be used in a genetic program for discovering technical trading rules. Neely [20] uses the following operations

1. Arithmetic operations: plus, minus, times, divide, norm, average, max, min, lag
2. Boolean operations: and, or, not, greater than, less than.
3. Conditional operations: if-then, if-then-else
4. Numerical constants
5. Boolean constants: true, false.

These operators are very powerful and can be combined to form effective rules. In addition Allen[21] restricts the depth of the trees to size 10. Neely also imposes a limit of depth 10 and size 100. As noted in the paper the trees generated have some redundant expressions. In addition it might be argued that the trees generated are difficult to understand. The issue of comprehensibility is important in that for significant insight to be gained into the rules that work, the GP must produce easily understandable rules. Understanding the rules is also important for people who use the rules. Money managers, for example, would have more confidence in using rules that they can understand. .

Extensive work has been done the issue of comprehensibility for decision trees. To certain extent one can equate comprehensibility with the simplicity or conciseness of the model, but consistency with the domain knowledge of the users who must comprehend the model is also an important factor [21]. If a user were to possess a chunk ([22], [23]) corresponding to a significant portion of the model, the comprehensibility of the model would be better than if the model contained the same 'size' portion for which the user had no corresponding chunk. For expression trees the possibility of simplification via subsumption relationships is also exists. While doing this by hand may not be practical for large complex trees, building domain-specific simplifier or using Mathematica or Maple to simplify equations would also be possible.

In addition, a number of studies ([24], [25]) it is argued that deeper trees are less comprehensible especially if they are binary. A complete binary tree of depth 10 would have 1024 nodes and would call for a lot of effort to extract meaning from it. It is important to note that the comprehensibility of a tree is also related to the returns it generates. Any additional complexity in the rule must generate equivalent increase in returns.

2.3.2 Overfitting avoidance

Another problem that is encountered in many machine learning and data mining techniques, including GP is overfitting. This can occur when the learned or evolved model fits the particulars of the training data overly well and consequently does not generalize to new unseen examples. There are basically 3 approaches to avoiding overfitting

1. Penalizing complexity or biasing towards simplicity
2. Limiting the number of models used
3. Using a validation data set

Penalizing complexity or biasing toward simplicity may involve post-pruning of models or generating and hypothesizing models in the order from simple to complex or searching in the space of solutions from general to specific and using some stopping criterion.

There have been a number of studies ([26], [27], [28]) where accuracy has not been reduced or has even been improved as a result of simplifying trees by pruning. There

have also been theoretical arguments in favor of what has sometimes been referred to as Occam's Razor, namely that simpler models have greater predictive power and lead to less generalization error ([29], [30]).

However, Domingos[31] argues against these theoretical arguments and cites numerous recent empirical studies where simpler models have underperformed. Following Jensen & Cohen [32], Domingos regards the number of models considered rather than the complexity of the models as leading to overfitting.

For GP, overfitting is often avoided by limiting the number of generations, and limiting the size of the population would also result in reducing the number of models considered. A validation data set can be used to directly test generalization errors and thus directly decide between different models. It can be used to cutoff search thus limiting the number of models considered. Such a cutoff can also prevent complexity when the search is biased from simple to complex.

2.4 Technical Analysis

Several centuries ago, traders of rice in Japan developed a method to track the price of the produce [33]. This took the form of marking different levels on a page as the prices rose and fell and was a primitive form of technical analysis. Modern technical analysis has added many more technical trading tools that are available to the analyst in his quest for price direction. In contrast to fundamental analysts who study the intrinsic reasons for the stock's rise or fall – profits, earning per share, market share,

recent news, etc to establish where they think a stock is headed, technical analyst use only historical price and volume information. They believe that all news, fundamental factors and market psychology are reflected in the price value. The technical analyst therefore uses historical price charts to make his decisions of when to get in and out of the stock. It is commonly referred to as market timing or charting.

2.4.1 Different types of indicators

As a quick tutorial of technical analysis we will introduce some technical analysis tools that we will be using in this study and some sample analysis.

2.4.1.1 Candlesticks

Figure 6. shows the monthly stock prices for approximately the past 6 years of the S&P500 index. Each stick represents the opening, closing, high and low price of the day. The bottom of the thin line represents the low price of the day, the top represents the high and the rectangle is formed using the opening and closing prices. A green rectangle means the closing price was higher than the opening and a red rectangle implies the closing was lower than opening. These lend themselves to a host of interesting patterns used by the technical analyst. For example the reversals in October 2000 was indicated by a ‘star’ pattern. This pattern, signified by a opening price that is close or equal to the opening price and there is considerable difference in the high and low prices indicates a reversal or indecision. There are more than 30 patterns that are based on combinations of the candlesticks alone.

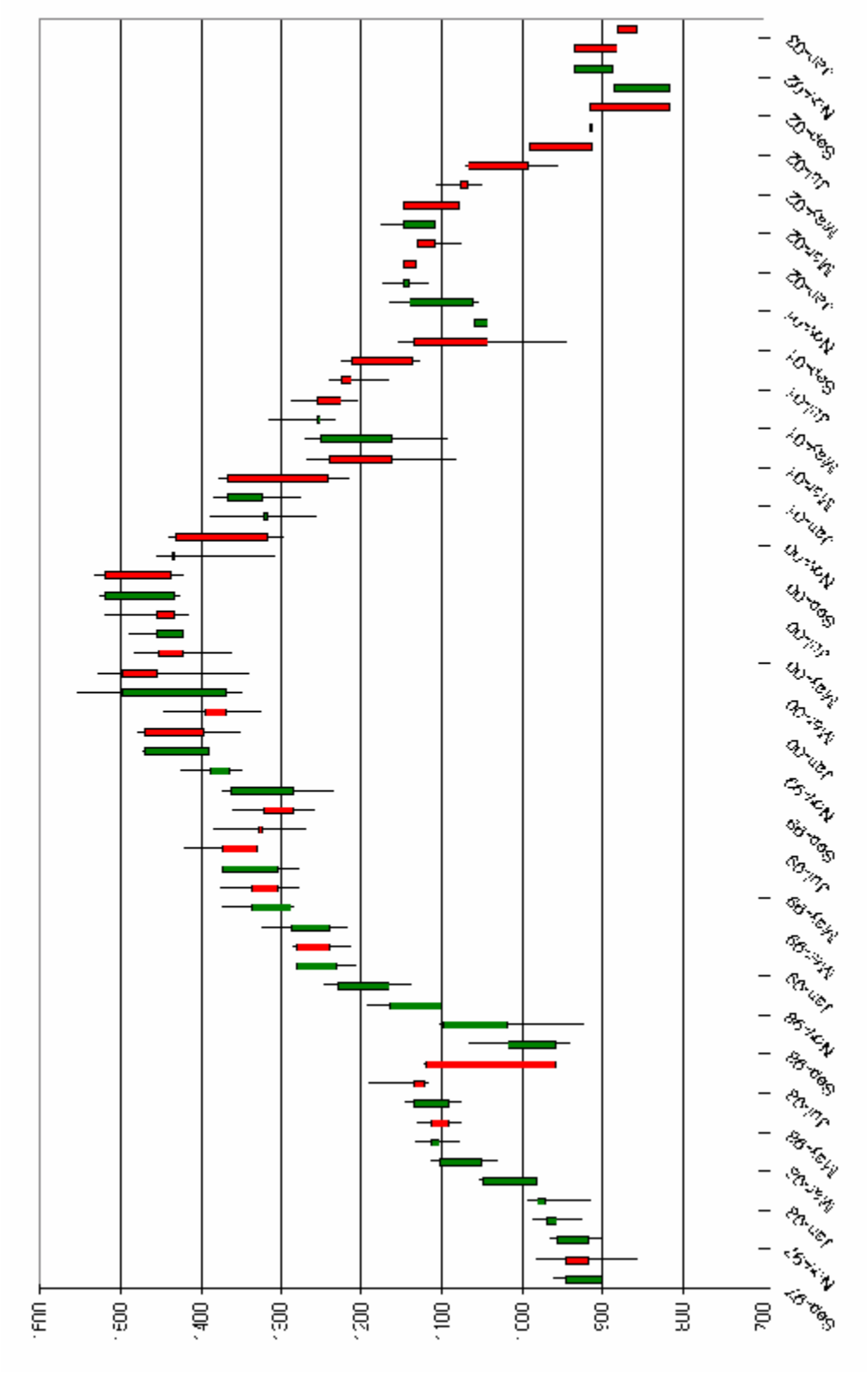


Figure 6: Technical Analysis Indicators – Volume and Japanese Candles

2.4.1.2 Moving Averages

More modern and one of the most commonly used tools by modern analysts is the moving average. For example in the graph below for IBM, the black line is the closing price for the month and the red and blue lines indicate the 3 month and 9 month moving average respectively. Moving averages are derived by averaging the price over past prices. The 3-month moving average in the chart below (Fig. 7) is the average of the current month and the past 2 months. Moving averages tend to smooth out the price curve and help visualize the long-term trend. For this reason they are often used in pairs, one to indicate much longer terms and another to indicate shorter term movements. They have the disadvantage of being a trend following signal instead of a trend predicting one and are often too late to be of use for short term signaling. Moving averages signal turnarounds when the price crosses over the moving average or if 2 moving averages of different periods intersect. If the moving average value is less than the price, it indicates that the market is bullish. The 9-month moving average (blue) is below the price level for the bullish period from 1994 to 200. Conversely the period from 2002 –2003 is identified as bearish by the price being below the moving average. Shorter term moving averages can be used to signal quicker movements (the period between March 1999 to January 2000).

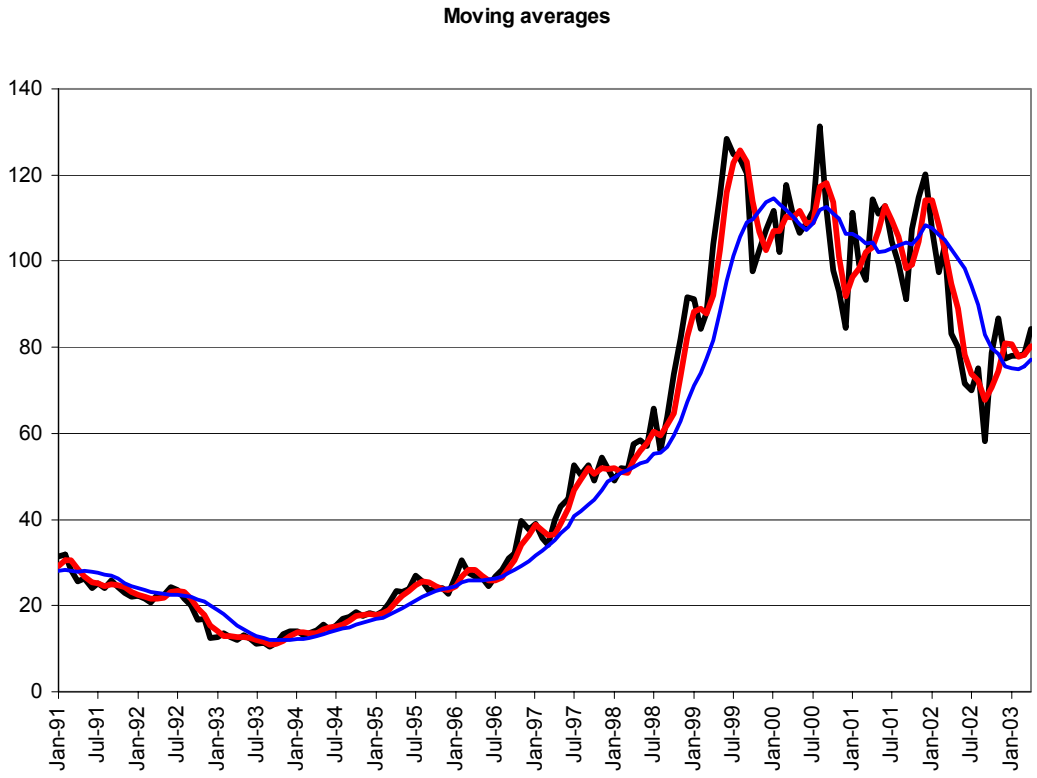


Figure 7: Technical Analysis Indicators – Moving Averages

2.4.1.3 Trend Lines

There is perhaps only one more technical tool that is easier and more widely used than the moving average. The trend line is visually appealing and easy to interpret. Trend lines are constructed by joining local maxima by a straight line and extending it to the present day. Such lines are upper resistance levels. Similar lower resistance levels can be drawn by connecting the local minima. In the above graph of IBM monthly prices (Fig. 8), 4 trend lines have been identified. The 2 blue lines identify lower resistance levels and the red and orange lines are upper resistance levels. As can be seen, prices face resistance at these levels and tend to move by oscillating in the channels created. Any breakout is followed by a large movement in the direction

of the breakout. It is therefore profitable to identify such breakouts. It can also be noted that after the breakout, resistance lines may become support lines (the red line) and vice-versa.

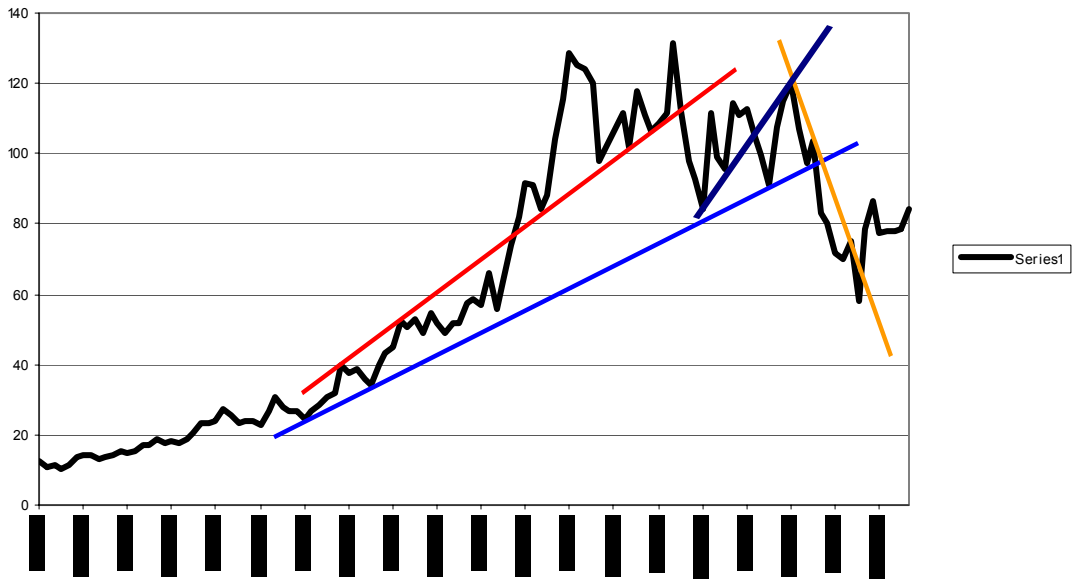


Figure 8: Technical Analysis Indicators – Trend Lines

2.4.1.4 Rate of Change

The rate of change is a momentum indicator, so called because it indicates the change in sentiment of the stock. The graph above (Fig. 9) compares the stock price with the 3-month (blue) and the 12-month (red) rate-of-change (ROC) indicator. The value for ROC is given by

$$ROCn = Close\ (current)-Close\ (current-n+1)*100/Close\ (current-n+1)$$

This gives the percentage change in the price for the period specified. The ROC is used to indicate overbought/undersold times in the market. For example, in July 93 the red line is at the lowest point indicating that a change in direction is soon to occur. Similarly the high of August 1999, is indicated by the consequent downtrend in the ROC. Most analysts use 2 ROC indicators, one for the short term and another for the longer term.

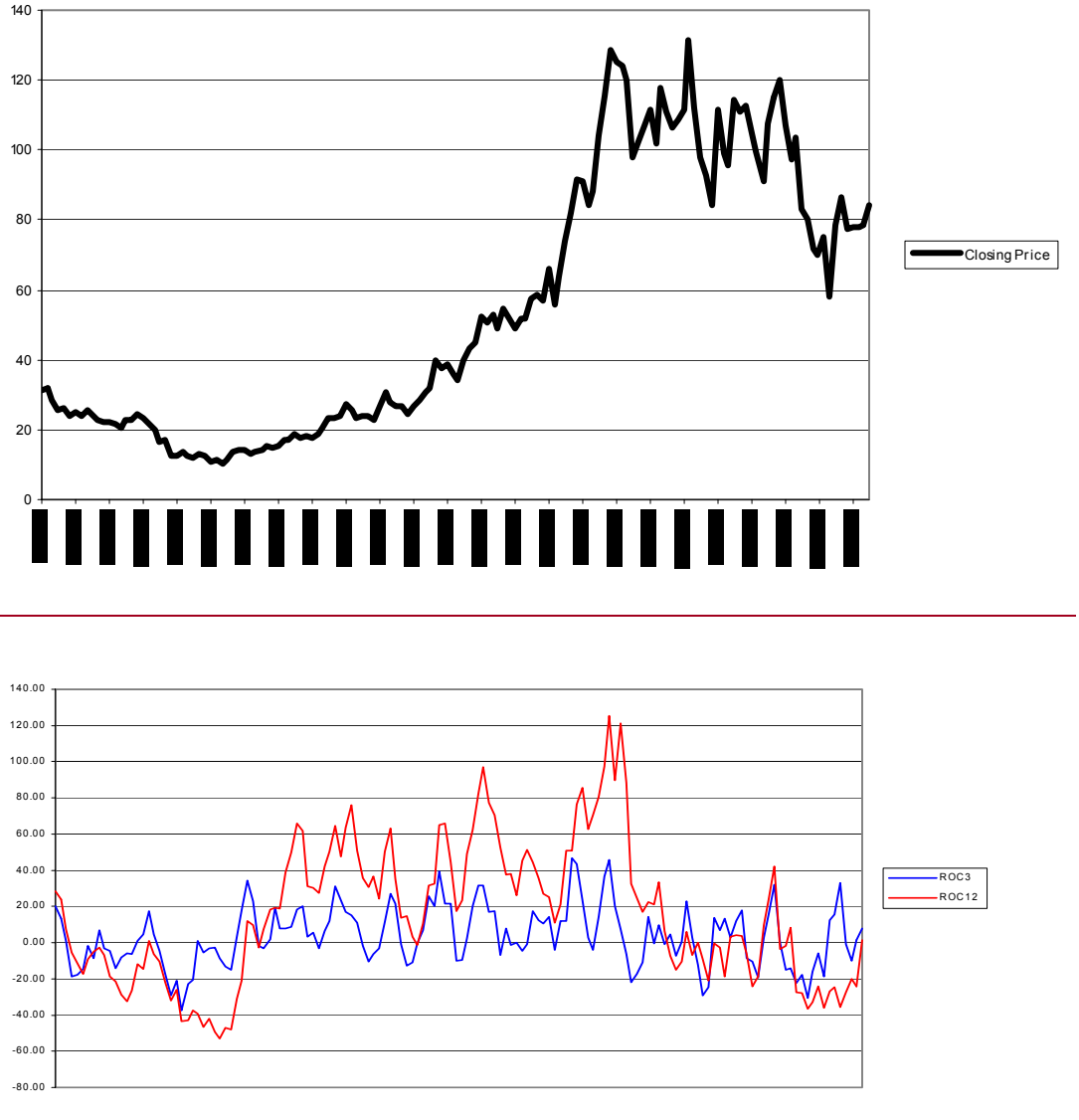


Figure 9: Technical Analysis Indicators – Rate of Change Indicators

2.4.1.5 Volume

Volume indicates the number of shares of a stock that have been traded. The strength of a trend is validated by the volume. In addition, periods of low volume may indicate uncertainty and a possible change in trend. Fig. 10 charts out the volume with the relation to the price. The peak in volume during Oct-1999 during the short fall of the price indicates sharp selling pressure and uncertainty in the new levels the stock price has reached.

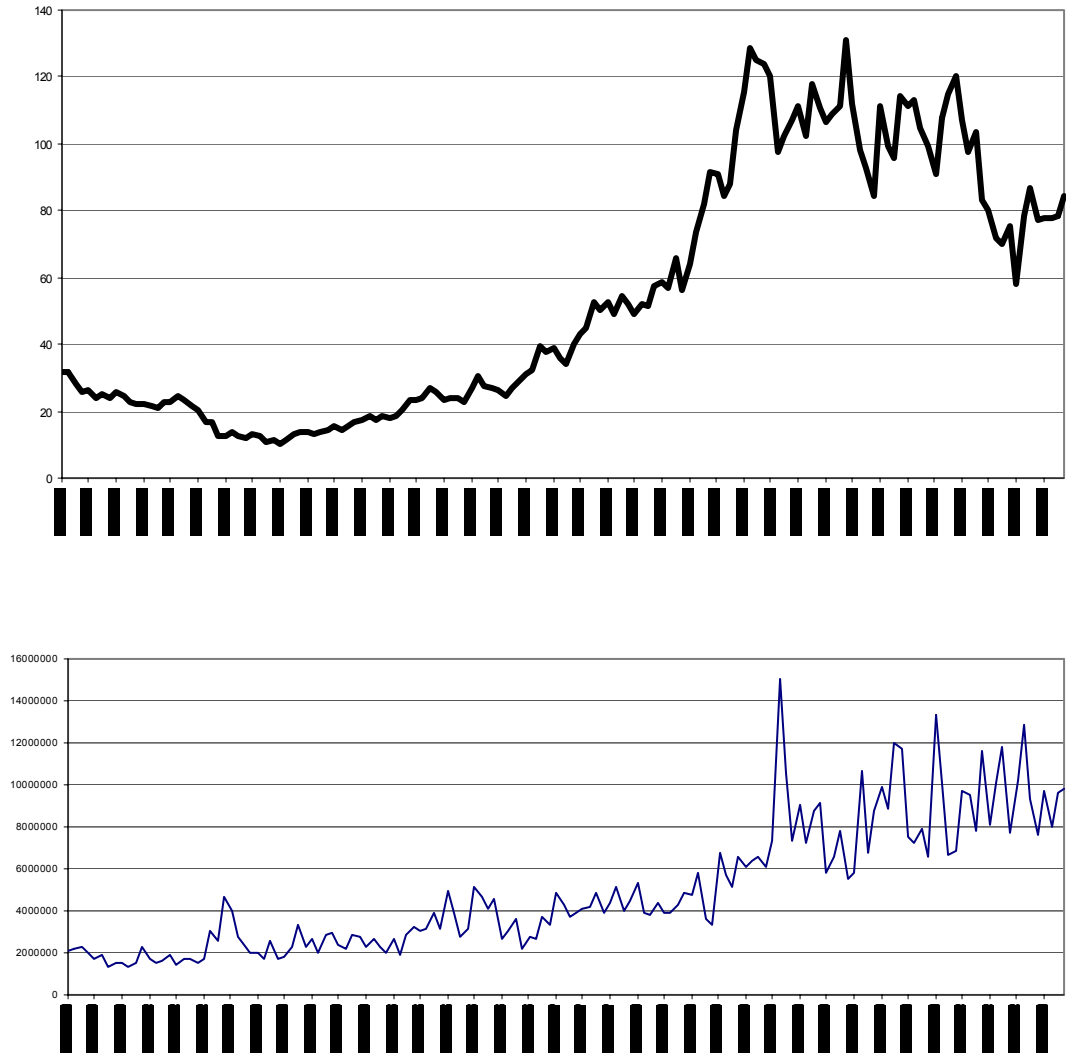


Figure 10 : Technical Analysis Indicators – Price and Volume

2.4.2 Work on technical analysis

There has been quite some work on technical analysis, the path breaking paper being [34] which shows that the profits generated by Dow Theory (the forerunner of technical analysis) are worse than buy and hold. This work has recently been challenged by [35] Brown et al. who show that the Dow Theory may have some merit after all. Fama [36] concludes that the statistical evidence for random walk over charting is overwhelming. Dennis [37] makes similar conclusions. Treynor and Ferguson[38] suggest that the value of technical analysis is in the efficient exploitation of non-price information. Their work is further discussed in [39]. In addition [40] Brown and Jennings make the case that technical analysis has value in markets where traders are looking for short-term gains. Franket and Froot [41] add a new dimension by suggesting that the market has periods in which it follows technical analysts and periods in which fundamental analysis has more weight. Neftci [42] uses moving averages to successfully trade on the Dow Jones Industrial. The existence of long term temporal dependencies is shown by Goetzamann [43].

As can be seen, there is no definite conclusion on whether technical analysis is profitable or to which markets it is applicable. Economists continue to believe in the efficient stock market hypothesis but are now considering different models that lead to price equilibrium. The technical analysts, on the other hand are still around Wall Street and the foreign exchange markets and large investment houses use their skills.

Is summary, most opponents of technical analysis and market timing make the following points

1. It is extremely difficult to do (if possible) and the competing buy-and-hold strategy is easier to use.
2. The transaction costs that accompany market-timing strategies must be made up.
3. All current knowledge of the stock is quickly and completely incorporated into the current price (the efficient market hypothesis)

Technical analysts on the other hand believe that

1. The prices reflect not just the fundamentals of a stock but also the psychology of the investors.
2. Market information is not as readily factored into the price as it is thought too.
3. Past prices give indications of future trends.

2.5 GP In Finance

Over the last 10 years there have been various publications of GA in finance but it is only since 1994 that GPs were beginning to be used for financial applications. A lot of this work has been in the area of forecasting and trading, which are closely related. The use of technical analysis has also been studied for forecasting. Technical analysis has been used to quite an extent for foreign exchange trading. This is partially because of the number of factors that affect the exchange rate and make it difficult to forecast well by conventional methods. Technical analysis provides a simpler

alternative. Neely [22] reports strong evidence of out-of-sample excess returns for technical trading rules over 6 exchange rates and ability to detect patterns that are not recognized by standard statistical models. There has also been work on the forecasting and trading of stock indices. The most important work here is Allen and Karjalainen's [21] who discover that the rules do not have consistent excess returns over a simple buy –and-hold strategy in the out-of-sample periods.

Allen and Karjalainen implement the GP with using the language described below

1. Arithmetic operations: plus, minus, times, divide, norm, average, max, min, lag
2. Boolean operations: and, or, not, greater than, less than.
3. Conditional operations: if-then, if-then-else
4. Numerical constants
5. Boolean constants: true, false.

Their fitness function calculates the excess return on the data over the buy-and-hold. A population of 500 trees is used in a 50-generation GP. The data is the S&P500 index daily prices. They guard against overfitting by using a validation data set which they obtain by splitting the data into 3 periods: a training period (5 years), a validation period (2 years) and a test period (the remainder). They train the data on the first period and after each generation apply the fittest rule (the one that has the largest excess return) on the validation period. If this rule does better on the validation period than any other rule so far encountered, it is saved as the best found so far.

Because of the nature of GPs, they end up with rules from about 5 to 10 levels deep that contain redundant sub trees and sub trees that are never processed. The choice of operators also renders the tree difficult to human interpretation and they lack an intuitive feel for the underlying significance. In order to measure the complexity of the rule generated, they compare the return of the rules generated to the returns generated by simpler rules formed of

1. Comparison operators (‘<’,’>’)
2. A varying time window
3. An operator on the time window: max, min and average.
4. Comparing normalized prices to a constant.

They conclude that the results obtained do not earn excess returns over a simple buy-and-hold strategy in the out of sample period if transaction costs are considered.

3 Innovations and Experimental Design

This chapter describes changes from previous work, especially Allen and Karjalainen [21]. We suggest changes and describe the experimental design and details for the experiments to test these changes. Details of the genome design, technical indicators used and data used are described.

3.1 Proposed Changes to the Allen and Karjalainen approach

To address the problems of overfitting and comprehensibility mentioned in chapter 2, four changes are proposed

3.1.1 Using a Complexity Penalizing Factor

We use a complexity-penalizing factor that reduces the fitness level of the tree as the complexity of the tree increases. It is possible to equate roughly the comprehensibility of the tree with its depth or its size. If there is, therefore, a small tree with the same return as a larger tree, the smaller tree is declared more fit. This has 2 effects – by restricting the growth of the tree to what is essential to increase return, we increase comprehensibility. Also the smaller tree has fewer nodes and is less likely to over fit the training data. By not restricting the size of the tree to an absolute value, and only encouraging it to be smaller, it is hoped that the gene pool remains rich and large. It also admits the possibility of finding a very profitable tree that is larger than the size

we are looking for. Such increased profitability would be of a nature that would allow a compromise to the comprehensibility of the tree.

There are also a number of different ways the factor can be constructed. [44] describes one of them. In the search for factors, it is important to note that the factor should be of a general nature. As an example, a subtractive factor was found to work well, but was not general enough to be used at different scales and was rejected. Finally a multiplicative scaling factor of the form

*Corrected fitness = actual fitness * desired-depth/max(actual-depth,desired-depth)*

was used.

As discussed above, reducing complexity of models has sometimes led to fewer and sometimes to more generalization errors. We therefore compare the performance of the models generated by GP with vs. without the complexity-penalizing factor.

3.1.2 Technical Trading Rule Expression Language

Most previous studies have used moving average crossovers, moving averages and trading range breakouts as the main technical indicators. It was decided to extend this set by using rate of change as volatility indicators, trend lines and support levels. The details of these indicators are discussed below. By using these indicators we are biasing the search, but are adding a large amount of comprehensibility because these blocks are well recognized and understood by the technical analyst. Of particular note are the trend line constructions.

Arithmetic operators such as (+, -, *, /) make the rule difficult to understand. For example in initial trial runs it was found that we obtain rules that contain sub-trees such as the one shown below (Fig. 11) that represents the rule

$$MA3*MA3 < Price*MA6$$

Such a rule that might be profitable does not indicate very well what the rule is trying to do.

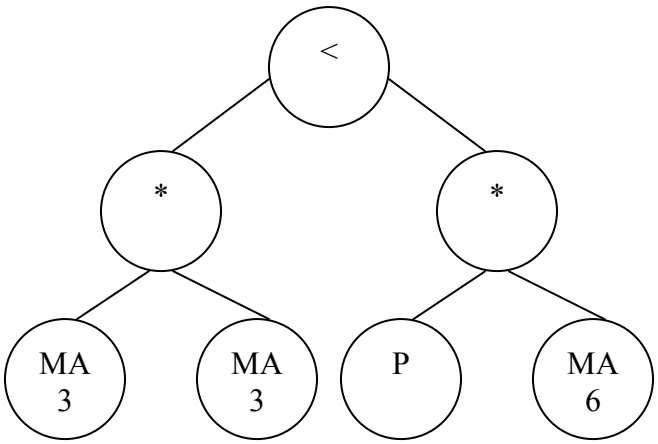


Figure 11: A rule that is difficult to comprehend

It was therefore decided to do away with arithmetic operators and leave only the following types of operators

a. Boolean operators: And, Or, Not

b. Comparison operators: <, >

c. Technical Analysis indicators (prices, moving averages, etc.) that give a number.

This reduces the number of operators and makes the structure less powerful, but it is hoped that with the judicious choice of technical indicators, this can be overcome. We have also done away with the usual constructs of "if-then-else" and the boolean constants true and false.

3.1.3 Fitness functions

A common fitness function for technical trading rule generation uses past return over the entire training period as the fitness function. In certain situations, this might result in the GP produced rule producing large returns but only over a small period of the training period. Such a rule would not generate consistent annual returns over a large period but erratic periods of high and low returns. In a desire to avoid this, we propose a fitness function that measures the number of years that the annual return is greater than or equal to the minimum of the risk free return and the stock market return for that year. We would expect such a function to provide more consistent returns over a longer period.

3.1.4 Co-evolution: Using Distinct Buy and Sell Rules

The problem of having a single rule decide when to get in and out of the market may perhaps be better solved by having 2 rules – one to decide when to get in and another to decide when to get out. This would result in specialization of buying and selling rules. These rules then would have to developed and paired together. This can be accomplished by coevolving the rules together. Essentially the structure and nodes comprising the tree remain the same and only the function assigned to each tree changes by this arrangement. Also because the rules are working together to attain a common goal (maximization of the number of positive periods or total return), the coevolution is of a cooperative nature.

As mentioned above in Chapter 2, this leads to issues with choosing collaborators and credit assignment. The credit assignment for this cooperative coevolution problem is easily solved by assigning the return generated by the pair to each. Equal credit is given to the buy and sell tree that worked together to generate the return.

The choosing of collaborators is a more complicated issue. We experiment with the different possible variations outlined above.

3.2 Data

3.2.1 Choice of data series

For data, historical monthly prices for the S&P 500 index were used. This index was chosen for the following reasons

1. It is the most commonly used benchmark used by investment professionals to measure the performance of their portfolios. Portfolios are evaluated by their ability to beat the S&P 500's return. It is therefore easy and meaningful to make comparisons to the profits generated by the rules.
2. The S&P 500 indicates the overall trend of the market and the economic climate. The companies included in this index fund are chosen to be representative of the important industries within the U.S. economy. This is borne out by the fact that the S&P 500 index is used by the U.S Department of Commerce as 1 of 11 leading economic indicators to signal potential turning points in the national economy. It

stands to reason that there has been more long term sustained interest in predicting the ups and downs of the S&P 500 than in most other stocks and funds

3. Splits and other capital corrections do not affect the continuity of the index. The index measures the amount of investment by taking the product of the price of a stock and the number of outstanding stocks. The sum is then related to the standard 1941-43 base, which is given a value of 10. This has the effect of ensuring continuity of the index and avoiding distortions introduced by splits and rights. Continuity helps the technical analyst a great deal by providing him a direct comparison to previous movements and values. We have however ignored the changes that might be introduced by reinvesting dividends.

4. Most previous research has been conducted on the S&P 500. It is easiest to make comparisons to these works if we used this index.

5. It is readily available and has a large amount of historical data that is easily available (for example http://finance.yahoo.com has quotes from at least 1955 onwards). This provides for a large amount of training data that is essential for machine learning techniques. Ready availability also makes it easier for other researchers to use.

3.2.2 Risk Free Interest Rates

Even though it would be possible to consider just the value of the scrip for training and comparison, it would not by itself be an accurate measure. The tradeoff between perceived risk/volatility and return is the defining factor that pushes investment in and out of the equity markets. As a measure of comparison, the risk free interest rates

available by buying government bonds and treasury bills form an important base level. In common with other researchers we have used the 3-month government treasury bills rates available at (http://research.stlouisfed.org/fred/data/irates/tb3ms).

3.2.3 Indicators Derived and Used from the S&P 500

The historical data we collected from the S&P 500 consists of

1. Opening price of the month
2. Closing price of the month
3. Highest price of the month
4. Lowest price of the month

We then went ahead to derive some technical indicators from this data. The choice of these indicators was based on 2 considerations. Firstly, that they be commonly used by technical analysts and easily understandable. Secondly there exist some previous work that indicates their usefulness. These criteria allowed us to choose

1. Moving averages and
2. Previous Maxima and Minima
3. Trend Lines or trading ranges

as focus areas. All 3 have been validated in previous studies [44]. For good measure we added 2 indicators that are widely used but not studied as well

1. Volatility represented by the Rate-of-Change or ROC indicator
2. Breakout gaps derived by including the previous month's close.

3.2.4 Moving Averages

We decided to use a combination of long-term, mid-term and short-term simple moving averages as advocated by most technical analysts. We spread it out as 10 month, 6 month, and 3 month averages. In preliminary runs we discovered that the short-term indicator could be shortened further without generating false signals and improving greatly the responsiveness of the rules. We therefore added a 2-month average as well. These were pre-calculated from the closing price of the months to avoid expensive re-computation every time they were needed.

$$MA_n(X) = [Close\ (X)+Close(X-1)+Close(X-2)+\ldots+Close(X-n)]/n$$

3.2.5 Previous Maxima and Minima

Local maxima and minima can be identified for months previous to the current. Because we are looking for significant resistance levels, minima and maxima are identified from the moving average values. This helps to weed out erratic price movements and smoothes out the curve to obtain values that are more significant. In order to draw a trend line for the trading range, we need at least 2 previous values of the maxima and/or minima. Consequently we find the previous maxima and the maxima prior to it and repeat the process for the minima

3.2.6 Trend Lines or trading ranges

Trend lines are one of the analyst's most used tools. They are drawn by connecting up 2 or more local minima or maxima. This results in support levels and resistance levels

respectively. It is possible to approximate this process by connecting the 2 previous minima/maxima. One obtains significantly better resistance levels by using the short term moving average to draw them instead of vanilla price values. Therefore the 2-month moving average values were used. It must be noted that using a number of different moving averages can generate a number of resistance levels and trend lines that might be useful. Longer term moving averages can result in longer term support and resistance values.

3.2.7 Rate of Change

As with the moving averages we use a short-term and a long-term measure of volatility – 3 month and 12 month. The values are pre-computed for computational efficiency.

$$ROC_n\,(X)=[Close\,(X)-Close\,(X-(n-1)]*100/Close\,(X-(n-1))$$

3.2.8 Partitioning the Data

The need for a 10-month moving average and a 12-month ROC value means the first year of the data is unusable for training. In addition we want the minima and maxima to be defined at all points, so we start the training period effectively at 1960. The data from 1960 to 2002 has to therefore be divided up into a training and test period. For the training to be significant we want the in-sample period to be at least twice as long as the out-of-sample period. We therefore decided to allocate 1960-1990 as the in sample and 1991-2002 as the out-of-sample period. Fig. 12 is the price chart for the monthly prices for this period.

Figure 12: Price chart of monthly closing values from 1960-1990

3.3 Details of the Genome Structure

The operators we use for the genome have been discussed above. Here we take a look at the details of the tree representation.

Most technical analysts use comparisons between two indicators to decide on interesting points in the market rather than a mathematical formula. Volatility indicators are sometimes an exception to this. They identify overbought/oversold values on a comparison to past values but also to some extent on the level of the indicator.

Consider the indicators we are using

1. Moving averages
2. Previous maxima and minima
3. Trend lines
4. Last month's price data.
5. Rate of change indicator
6. Volume of current month and previous month

With the exception of rate of change and volume, all the other indicators give us values that are comparable to the current price or on the same scale. This means they can all be compared to each other. This is very much what the chartist does. He

compares moving averages, maxima/minima, trend lines, last month's data and then takes into consideration a separate graph of the rate of change and another graph of the volume. For the first set of indicators the comparisons made are mostly of the nature of crossovers. They wait for 2 values to crossover, for example the current price is compared to the trendline that currently exists. If the price goes through trendline (crossover) it signals an important point in the market. This is why we hope that using just the comparison operators and avoiding arithmetic operators should prove useful.

For the volume and rate of change indicators, we again take recourse to the comparison operators, but here the patterns are not as easy to discover. For example one could use moving averages of the volume for crossovers and for the rate of change indicators, it might be worthwhile using 2 different periods and looking for crossovers thought admittedly this is a rather poor way of using these 2 indicators. We therefore just restrict ourselves to the ROC-3 months, ROC – 12 months. And use this month's volume and last month's volume. One might argue that the volume increase is used to substantiate a price movement and for this purpose, comparison of the current month's volume to the previous month's should serve the purpose.

Having decided therefore to use only comparison operators, the basic structure of the rules becomes evident. For comparison operators, we use the < and > operators. The simplest tree/rule would therefore be of the form of Fig.13

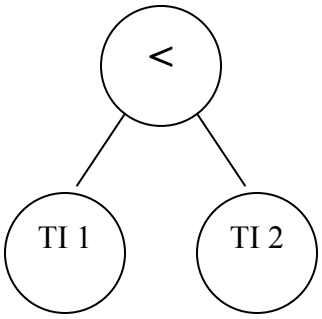


Figure 13: The simplest tree

where TI-1 and TI-2 indicate 2 technical indicators. For example TI-1 could be MA3 and TI-2 could be MA6. In that case, the rule would be

$$MA3 < MA6$$

Of course, if TI-1 were this month's volume (V) then TI-2 must be last month's volume (v). Similarly if ROC12 is TI-1 or TI-2, the other TI must be ROC3.

Now, to generate more complex trading rules, we need to combine these simple rules using boolean operators such as AND, OR and NOT. For example we could have two rules (P > MA6) and (MA3 < MA6) put together with either an AND (below) or an OR. The resulting structure would look like Fig.14

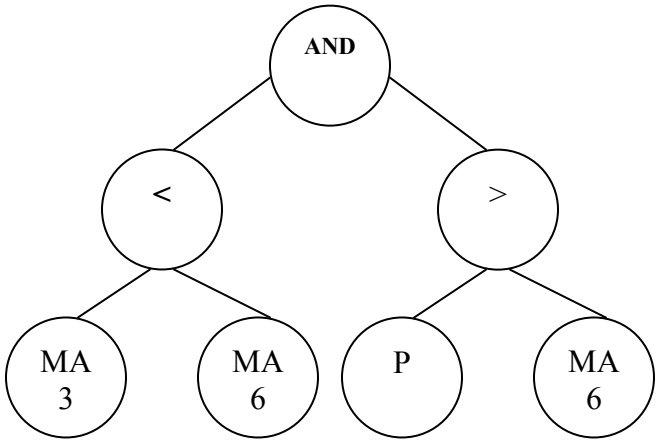


Figure 14: How the Boolean operator fits in

This structure can be extended by using more boolean operators to create more complicated structures as in Fig. 15.

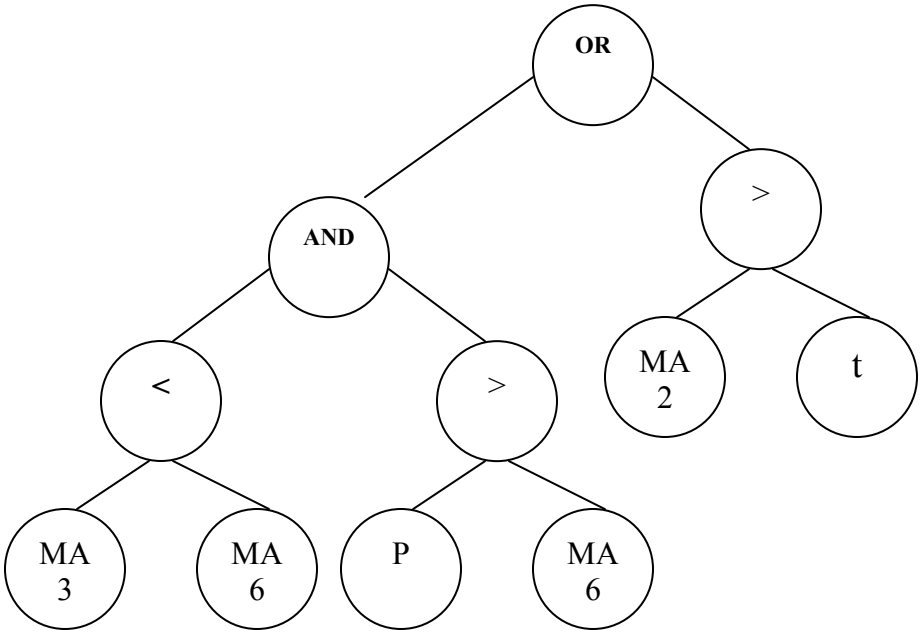


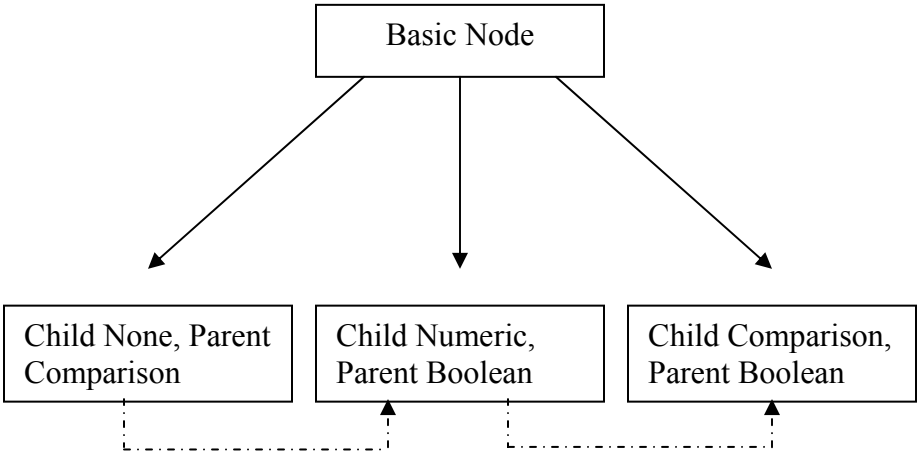
Figure 15: A larger tree

This results in trees in which the 2 leaf nodes and the node connecting it form a simple rule and all the nodes above it put them together. These 3 nodes process real number values and return boolean (true/false) results that are further combined by the nodes above it. The entire tree therefore always returns a True or False value for a particular month.

3.4 Genetic Program Parameters

3.4.1 Types of Nodes

As defined above, the nodes are implemented as a class hierarchy of different node types, identified by the types of children they can have and the parents that can have them



Child None, Parent Comparison: Technical Indicators

Child Numeric, Parent Boolean: <, >

Child Comparison, Parent Boolean: AND, OR, NOT

Figure 16: Hierarchy of nodes

This relationship is described graphically in Fig. 16. The dotted lines indicate the child → parent relationship.

3.4.2 Random Tree Initialization

The initial random trees are generated such that they have a maximum depth of 10. A depth-first recursive initialization function is used. The algorithm is as follows and is started by calling the function with a depth 0.

Initialize Tree (depth, Type_of_parent_node)

1. If (depth==0) OR (parent node == Boolean)
    a. Randomly choose comparison or boolean operator as first node

   Else if (depth==10) /*max depth*/ OR (parent node == comparison)
    a. Choose a technical indicator
2. Call Initialize Tree for left node (depth+1, current node)
3. If (Boolean operator! = NOT)
    a. Call Initialize Tree for right node (depth+1,current node)

3.4.3 Mutation Operator

For the mutation operation we have a few choices that are detailed in the GALib[45] package, written by Matthew Wall at the Massachusetts Institute of Technology that we used.

1. Sub-tree Swap Mutation: We could take a sub tree and graft it elsewhere in the tree

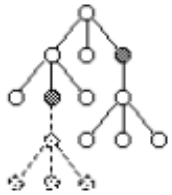


Figure 17: Sub tree swap mutation

This cannot be used in this case because as can be seen, it violates the parent-child node definitions. Specifically, it would end up with the left hand side dark node with a hanging boolean/comparison operator and would give the right hand side dark node (which is a technical indicator and cannot have children) children.

2. Tree Node Swap Mutation: Here we swap 2 nodes. The same problem as above exists. Of course here we can check the nodes swapped to ensure that they are of the same type before we do it.

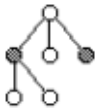


Figure 18 : Tree Node Swap Mutation

3. Sub-Tree Destructive Mutation: This method destroys the sub tree of a random tree node. This again has the same problem as the sub-tree swap mutation, resulting in hanging boolean/comparison operators.

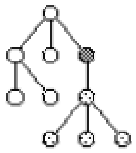


Figure 19 : Sub Tree Destructive Mutation

4. We use a fourth option. We choose a random node in the tree and then replace it with a similar node of a different type. For example if the node we chose was a boolean operator, say AND, we could change it to OR. Similarly, '<' could be changed to '>'. Technical indicators like MA3 can be changed to MA6. We must ensure however that for nodes like Rate-of-Change we don't change the node to Price or Volume that makes the comparison meaningless.

3.4.4 Crossover Operation

The crossover operation is rather straightforward and easy to implement. As shown in the figure below, a random node is chosen from one of the 2 parents (say mom) and a random node is chosen from dad. These sub trees are then swapped.

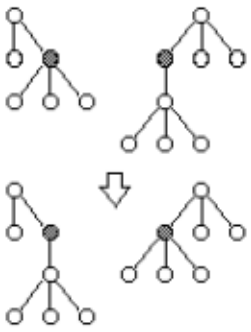


Figure 20: Single Point Crossover

Here as well, one must watch out for the types of nodes chosen. They must be of the same type for meaningful offspring. The algorithm is as follows

1. Choose mom, dad on the basis of fitness
2. For mom, choose a random node and identify the type
3. For dad choose a random node and identify the type.
4. If the node types are different go back to 2.
5. Perform the crossover.

3.4.5 The Objective Function

The objective function uses the technical trading rule to carry out trades to achieve the required objective. Initially, we start out by attempting to maximize the total return over the training period on a 1000$ investment. The algorithm is

1. Initialize
    a. Month = 1. Year = 1960.
    b. Buy = true.
    c. InTheMarket = false
    d. Amount =1000
2. While period is not over
    a. If (Buy==true && not (InTheMarket)) buy into the market
    b. Else if (Buy==false && InTheMarket) sell
    c. If (Not (InTheMarket)) increment Amount by risk-free interest rate.
3. Return the final Amount.

The action matrix for the tree can be described as in the following table

Table 1: Action matrix for single rule tree

	True	**False**
In the market	Do Nothing	Sell
Out of the market	Buy	Do Nothing

Transaction costs are used for each transaction (buy/sell). The standard transaction costs of 0.5% and 0.25% are used.

3.5 Experiments

In this section we describe the hypotheses to be tested and the experiments suggested to test them. These tests include the effect of the complexity penalizing factor, the effect of choosing a different evaluation function and different co-evolution strategies.

3.5.1 Experiment – I (Effect of complexity-penalizing factor)

As mentioned in Chapter 2, there have been some results suggesting that reducing the size of the tree may or may not decrease the performance of the tree. Our first experiment is therefore to measure the effect of the complexity-penalizing factor on performance.

We therefore run the GP, with the factor and without and study the results. We use the following parameters

1. Steady – state GP
2. Number of generations: 100
3. Population size: 150 trees
4. Crossover rate: 0.8
5. Mutation rate: 0.1
6. Half the population is replaced at each generation. The half replaced represents the worst half.
7. Training data: Jan 1960 – Dec 1990
8. Test data: Jan 1991 – Dec 2002
9. Objective function
    a. Value = Total return on initial investment
    b. Transaction cost = 0.5%
    c. Risk-free Interest rate = 3 Month T-bill
10. Factor used in one set of runs: objective score * (5/(max (TreeDepth, 5))

The best tree at the end of the run is saved and run on the out of sample data. We run this tree on the out-of-sample data. A total of 10 runs, each starting with a different seed is made. This ensures that the initial populations of both sets of runs are the same.

3.5.2 Experiment – II (Different Evaluation function)

From the results of Experiment –I it was noted that some of the rules generated smaller returns on the test period as the returns on the training period improved. It was thought that this might be because of a tendency to extract maximum profit from the training period but not necessarily in a consistent manner throughout the period. Also, an inordinate amount of weight is given to initial years in the training period because the returns generated there are carried over to investments in the latter part of the training period and even though rules may do well at the end of the period, they may not make enough back to compensate for rules that do well at the start.

Therefore a different evaluation function was proposed. This function would work similarly to the one of Experiment – I but with a couple of changes

1. The amount in hand or in the market is reset to 1000 at the end of the year
2. The fitness value is defined not as the return but as the number of years in which the rule gave returns that were better or equal to both the buy and hold strategy return for that year and the risk free interest return for that year.

To verify the hypothesis that this function yields better and more consistently performing trees, we ran 10 runs of each and compared the results. The parameters for the run were:

1. Steady – state GP
2. Number of generations: 150
3. Population size: 150 trees

4. Crossover rate: 0.8
5. Mutation rate: 0.1
6. Half the population is replaced at each generation. The half replaced represents the worst half.
7. Training data: Jan 1960 – Dec 1990
8. Test data: Jan 1991 – Dec 2002
9. Objective function
    a. Value
        i. Total return on initial investment for one set
        ii. Number of years the return is greater/equal to the maximum of risk-free and buy and hold strategies.
    b. Transaction cost = 0.5%
    c. Risk-free Interest rate = 3 Month T-bill
10. Factor used in both sets of runs: objective score * (5/(max (TreeDepth, 5))

The best tree at the end of the run is saved and run on the out of sample data. We run this tree on the out-of-sample data. A total of 10 runs, each starting with a different seed is made. This ensures that the initial populations of both sets of runs are the same.

3.5.3 Experiment – III (Using Paired buy and sell rules)

In order to test whether using two trees – one specialized for buying and one for selling improves performance we decided to use 2 trees. For this purpose we used paired buy-sell pairs as the population. Each species (buy and sell) evolves separately. This affects the crossover operator. Crossovers can now only be carried out between two similar sub trees (buy-to-buy, sell-to-sell). This ensures that the 2 sub trees have different gene pools that encourage their respective functions.

The tree structure remains the same as for Experiment-I but a for a small change. We introduce a combination node at the root that combines the two sub trees below it as shown in Fig.21. The left sub tree is designated as the buy tree and the right sub tree as the sell tree. The combination node can be designed to work in a couple of ways.

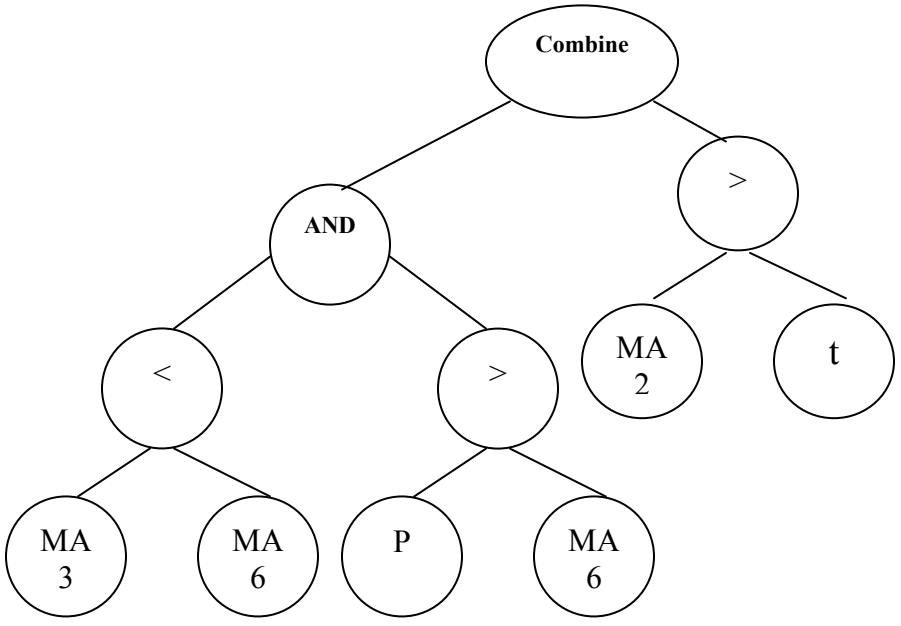


Figure 21: Structure of a paired tree

Consider for example the following action table

Table 2: Action matrix for paired rule - 1

MARKET	BUY RULE	SELL RULE	ACTION
In	True	True	Do Nothing
In	True	False	Do Nothing
In	False	True	SELL
In	False	False	Do Nothing
Out	True	True	Do Nothing
Out	True	False	BUY
Out	False	True	Do Nothing
Out	False	False	Do Nothing

This represents the condition that the stock is bought only when

1. Buy rule is true and Sell rule is false

And the stock is sold only when

2. Buy rule is false and Sell rule is true

For this rule the combination node would work as an XOR node. So

Buy → (Buy rule XOR Sell rule) AND (Buy rule = true)

Sell → (Buy rule XOR Sell rule) AND (Sell rule = true)

This method has the advantage of obtaining a very clear signal and confirmation from both rules for action. On the other hand it is difficult to generate successful rules because the buy rule has to know about the sell rule and vice versa to develop

effectively. In a way this is also equivalent to a voting scheme of 2 rules that can be developed from this pair by inverting the truth-value of the sell rule. Therefore we use an alternate arrangement where we just consider the buy rule for buying decisions and the sell rule for the selling decisions. Effectively we decide which one to consult depending on the current state and ignore the other tree at that time.

Table 3: Action matrix for paired rule - 2

MARKET	BUY RULE	SELL RULE	ACTION
In	X	True	SELL
In	X	False	Do Nothing
Out	True	X	Buy
Out	False	X	Do Nothing

In order to compare the performance of the co-evolved tree with the single rule tree we need to run both GPs and compare the results. From the results of Experiment-I, we know that the complexity-penalizing factor works and so we use the factor in the paired configuration and the single rule tree.

Once again we set the parameters as pretty much the same as above and

1. Steady – state GP
2. Number of generations: 150
3. Population size: 150 trees
4. Crossover rate: 0.8

5. Mutation rate: 0.1
6. Half the population is replaced at each generation. The half replaced represents the worst half.
7. Training data: Jan 1960 – Dec 1990
8. Test data: Jan 1991 – Dec 2002
9. Objective function
    d. Value = Number of years the return is greater/equal to the maximum of risk-free and buy and hold strategies.
    e. Transaction cost = 0.5%
    f. Risk-free Interest rate = 3 Month T-bill
10. Factor used in both: objective score * (5/(max (TreeDepth, 5))

The best tree at the end of the run is saved and run on the out of sample data. We run this tree on the out-of-sample data. A total of 10 runs, each starting with a different seed is made. In this case the initial trees produced are not the same.

3.5.4 Experiment – IV (Different types of co-evolution)

Experiment IV was conducted to evaluate different co-evolution strategies and compare them to the paired tree. Section 2.2 describes the different ways collaborators can be chosen from the other species. Consider Fig. 22 below:

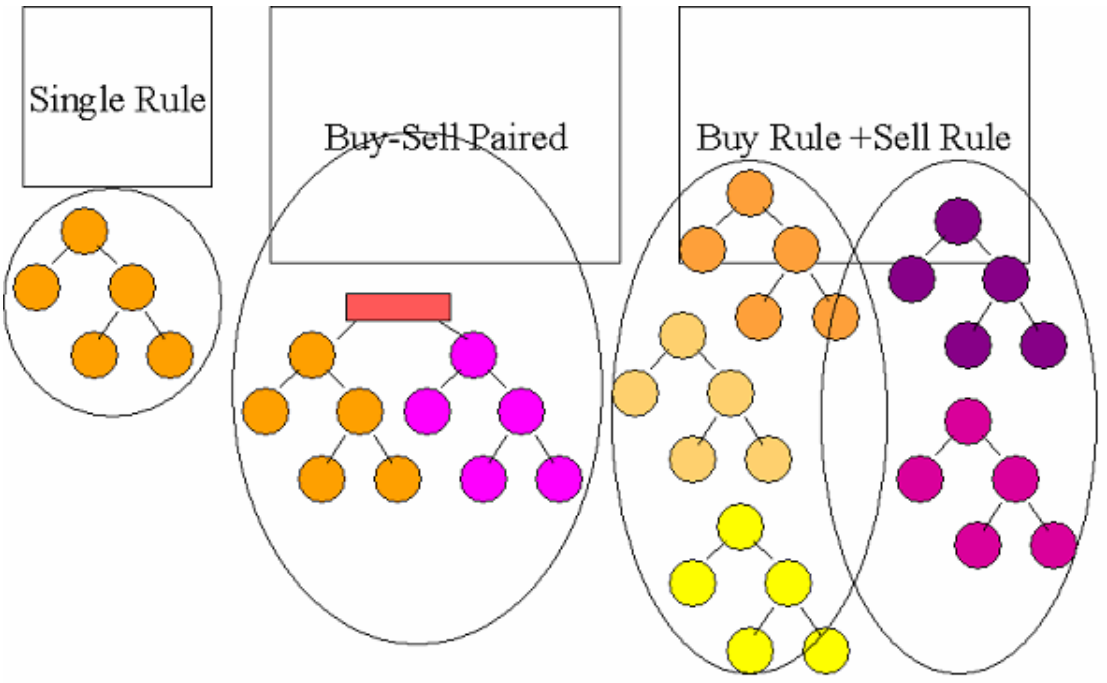


Figure 22: Overview of co-evolution strategies

In Experiment – III we compared the results of the Single Rule and the Buy-Sell Paired. Now we consider separately evolving populations of buy and sell rules where collaborators have to be chosen for evaluating the performance of the trees. Of the various possibilities mentioned we use the following 2 schemes

1. Choosing the 5 best collaborators from the other species
2. Choosing 5 random collaborators from the other species

For example, for evaluating the buy trees, we choose the best 5 sell trees based on the evaluation function. For the first generation, when fitness values have not yet been assigned we choose random collaborators. Then each buy tree is evaluated with all

the 5 sell trees and the average value is assigned to it. The process is repeated for the sell trees. Evaluating the trees with different collaborators may help in isolating a tree from the ill effects of a bad collaborator. For example, in a paired configuration a well performing buy tree may be paired with an ill –performing sell tree and will not get a chance to get credit for its 'goodness'. Choosing a variety of collaborators may minimize this effect.

In addition because trees that are similar to each other tend to have the same fitness levels, choosing the best 5 may result in similar trees. To avoid this we have also tested choosing random collaborators. Another variation to avoid this crowding is to replace all the trees except one instead of just half the population at every step.

The hypothesis is therefore, that the unpaired co-evolving trees are better than the paired trees. This is tested by running 3 sets:

1. Best 5 co-evolved
2. Random 5 co-evolved
3. Paired buy-sell.

The parameters are:

1. Steady – state GP
2. Number of generations: 150
3. Population size: 150 trees

4. Crossover rate: 0.8
5. Mutation rate: 0.1
6. Population replacement
    a. Half the population is replaced at each generation for the paired trees. The half replaced represents the worst half.
    b. All but one replaced for both species in separately coevolving GP. The ones replaced are the ones with the worst fitness.
7. Training data: Jan 1960 – Dec 1990
8. Test data: Jan 1991 – Dec 2002
9. Objective function
    a. Value = Number of years the return is greater/equal to the maximum of risk-free and buy and hold strategies.
    b. Transaction cost = 0.5%
    c. Risk-free Interest rate = 3 Month T-bill
10 Factor used in both: objective score * (5/(max (TreeDepth, 5))

The best tree at the end of the run is saved and run on the out of sample data. We run this tree on the out-of-sample data. A total of 10 runs, each starting with a different seed is made. In this case the initial trees produced are the same for the BEST5 and RANDOM5 runs.

4 Results

This chapter describes the results of the experiments described in chapter 3. The results are grouped by experiment number and the conclusions of each experiment are indicated. Conclusions are described in greater detail in Chapter 5.

4.1.1 Experiment-I (Effect of complexity-penalizing factor)

The 10 runs without the factor yielded the following results

Table 4: Results for experiment I -1

Without Factor			
Size	Depth	1960-1990	1991-2002
224	19	15195	3491
542	88	13856	3300
530	45	20265	1469
41	11	16832	2023
200	38	12818	1835
967	87	14984	1834
178	28	19605	1626
45	11	16367	1977
111	20	17637	1764
596	68	23702	2089
343	*42*	*17126*	*2141*

In comparison to the buy-and-hold strategy

Table 5: Result for Buy and hold strategy

	1960-1990	1991-2002
Buy&Hold	5457	2638

The entries in green indicate the values that beat the buy-and-hold. The last line of Table 4 indicates the average of the size, depth, in-sample return and the out-of-sample return respectively.

It is easily seen that the complexity of the tree is indeed a problem. The smallest tree has a depth of 11 and size 41 and the largest are almost a 1000 nodes large with a depth of approximately 90. This also has an effect on predictive performance. Even though the trees perform on average about 3 times as well on the in-sample data. This does not carry over to the out-of-sample data where the performance is worse than buy-and-hold. In fact, only 2 of the rules discovered beat the buy-and-hold in the out-of-sample data.

The results for the runs with the complexity-penalizing factor are

Table 6: Results for experiment I - 2

With Factor			
Size	**Depth**	**1960-1990**	**1991-2002**
15	5	10976	3128
15	5	13690	2006
3	2	8762	3377
15	5	13981	2003
3	2	8762	3377
12	5	14788	1685
15	4	15078	2096
3	2	8762	3377
15	5	9128	3697
3	2	8762	3377
10	***4***	***11269***	***2812***

Once again, the last row indicates the average of the column. Compared to Table 4, the results seem much more promising. The average for the in sample data is only twice as much as buy-and-hold but is indicative of the fact that it has not over fit the data. This is evident from the out-of-sample average that beats the buy-and-hold indicating that the main trends in the data have been identified and overfitting has

been considerably reduced. The results are significant to about 96%. The average size and depth also indicate very clearly that the rule is much more comprehensible. Of the 10 runs, 6 result in rules that beat the buy-and-hold.

These results clearly bear out the hypothesis that the addition of a complexity-penalizing factor increases both, the comprehensibility and reduce overfitting.

4.1.2 Experiment-II (Different Evaluation function)

In this section we study the results of running the GP with the newly proposed evaluation function that measures the number of years the GP produced rule beats the buy-and-hold and the risk-free return.

The 10 runs with the new evaluation function return the following values

Table 7 Results for experiment 2

New evaluation fn.	
1960-1990	1991-2002
7705	2064
7541	2166
7143	2654
7143	2654
6916	2953
7968	2982
7131	3184
7644	3437
7110	3691
8126	3983
7442.7	2976.8

Once again, the last row represents the averages. As can be seen, the average for the out-of-sample period has increased. More importantly, the results are consistently better. Only 2/10 of the rules have a performance worse than the buy-and-hold as compared to 4/10 in table 6.

We can conclude that this evaluation function does not do any worse than the first evaluation function. If anything it is only better.

4.1.3 Experiment-III (Using Paired buy and sell rules)

Here we compare the results of using a specialized buy and a specialized sell tree. The results for the 10 runs of the paired rule follow:

Table 8: Results for experiment III

Paired Rule	
1960-1990	1991-2002
10816	2807
12374	2823
11818	2837
19861	2856
12667	2911
11530	3258
22074	3434
18467	3475
14957	3476
10646	3541
14521	3141.8

Comparing the results with table 7 we see significant improvement in the average for the out-of-sample period and the in-sample period. In addition, we note that all the rules produced were consistently better than the buy-and-hold. The rule beats the buy-and-hold with a statistical significance of 99%.

We conclude that the specialization of function obtained by splitting the single rule into two rules certainly helped the predictive power and the data fitting capability.

4.1.4 Experiment –IV (Different types of co-evolution)

The results of Experiment-III suggested that we try other methods of coevolution, different from fixed pair coevolution. The results of the runs for using 5 random collaborators from the other species for coevolution are tabulated below

Table 9: Results for experiment IV - 1

Buy + Sell	
Random 5	
1960-1990	1991-2002
12757	2438
13832	2742
12911	2807
15243	2807
14747	2837
12077	2842
16249	3059
16117	3374
17746	3376
6233	3876
13791.2	3015.8

The average for the out-of-sample period beats the buy-and-hold and only one rule does worse than it. The results are statistically significant at 99% for beating the buy

and hold. In comparison with the paired rule results (Table 8), we don't find much difference.

For the runs with the Best 5, we have the following results

Table 10: Results for experiment 1V –2

Buy + Sell	
Best 5	
1960-1990	1991-2002
18898	2399
18795	2527
21525	2567
14525	2807
12893	2807
14819	2837
14809	2971
14377	3036
12893	3222
13820	3256
15735.4	2842.9

As can be seen, these perform somewhat better on the in sample. They still manage to beat the buy-and-hold on average but not as consistently. It might be conjectured that these rules are over fit to the training data.

5 Conclusions & Future Work

This chapter outlines the conclusions from the results of the experiments and indicates directions for future work.

5.1 Conclusions

This thesis has shown that it certainly seems possible to beat the buy-and-hold strategy of the S&P500 consistently using GP even when considering the rather high transaction costs of 0.5%. This is at odds to previous work where the GP produced rules do not consistently beat the S&P500. This is significant because it suggests that the markets may not as efficient as they are thought to be and it proves that there is value in technical analysis.

In addition, it has been shown that reducing the size of the produced rule does not affect performance adversely, but instead, by reducing overfitting actually enhances it. In other words, for this application domain at least, simplicity in the size of the description improves predictive ability.

We have also successfully shown that it is indeed possible to generate comprehensible trading rules that outperform buy and hold strategies. This might be an indication that the use of arithmetic operators adds too much complexity to be helpful and that the choice of operators is an important consideration.

The choice of technical indicators was justified because a large number of generated rules used the trend line indicators, the moving averages and previous maxima and minima.

We have also seen that the proposed annual evaluation function might be a better alternative to the usually used vanilla return maximization function. This function generates consistently better performance over a variety of different types of years. This is also important in building investor confidence. For example, a rule that made a large profit in a 10 year period but lost money consistently in the first 5 years would perhaps lose the investor's trust in the first 5-year period and not be used over the 10-year period to make up for the loss. Consistent and frequent positive returns are important in developing this trust. This is similar in some sense to the evaluation of money managers, whose performance is measured by how many years they have beat the market in the past.

By separating the buy and sell rules and evolving them separately, we obtain much better and certainly more consistent performance. In the comparison between the paired rule, random 5 and best 5 methods, there was not much difference but the results seem to indicate that the paired rules perform the best. They beat the buy-and-hold strategy with 99% significance. It appears that in the tradeoff between diversity of collaborators and the compatibility of a fixed (set of) collaborator(s), compatibility may be more important in this domain. This technique may be applicable to other

domains that are traditionally thought of as having a single undifferentiated solution. For such domains at least, one should consider using fixed collaborators from the other species for fitness evaluation.

5.2 Future Work

These results suggest various further directions for research:

Staged Priming

In the co-evolution schemes, the initial trees we consider are random trees. Instead we might perhaps use the rules generated by one of the schemes to jump-start the other ones. For example we could use the rules generated by the paired rules initializing the evolution of the Best 5 or Random 5 schemes or vice versa.

Combined Best +Random

We have considered a couple of schemes for co-evolution where the choice of collaborators has been Best 5 and Random 5. It is possible to use a best-of both-worlds strategy by using some best and some random trees as collaborators. This might improve performance and maintain diversity.

Competitive Coevolution

An intriguing possibility is to use competitive coevolution to evolve the technical trading rules. In a manner similar to Hillis' [13] work on sorting networks we could

have two species: one of rules and the other consisting of a block of years from the training period. Then as the rules evolved to perform well on the set of years, the set of years would evolve to become more difficult to perform on. This would lead to a rule that did well on a variety of years and this might carry over to the unseen testing sample.

Voting schemes

As an alternative to choosing rules and as suggested by various studies, it might be profitable to attempt a voting scheme where different rules vote on the decision to be taken. This is in a sense similar to diversification of a portfolio of rules that reduces the risk involved.

Appendix

Approximate Calculation of number of trees

For depth=2

There are 13 indicators apart from the rate-of-change and volume indicators that can be directly compared. Not counting the ROC and volume indicators, we have 13C2 ways of choosing 2 operators and one of 2 comparison operators to choose ('<','>'). Therefore for a tree of depth 2 we have at least **156** possible trees.

For depth = n (> 2)

For trees of depth > 2, the last 2 levels are made up of 2 technical indicators and a comparison operators. Also because there are 2^(n-2) such triples and 156 such possible triples. The last 2 levels contain 156 ^ (2^(n-2)). The nodes above them are made up of Boolean operators (AND, OR, NOT) and there are [2^(n-2)-1] such nodes. Ignoring NOT which has only one subtree, there are 2^[2^(n-2)-1] combinations for boolean nodes. The total number of combinations is therefore

[156^(2^(n-2))] * [2^[2^(n-2)-1]]

For the values of n=2 to 5 we have.

Depth	Number of possible trees
2	156
3	48672
4	4737927168
5	44895907698545000000

It must be noted that to find all trees with maximum depth 5 we have to sum all number of possible trees from n=2 to n=5. The large numbers of trees indicate clearly that GP will help find an efficient solution quickly and cheaply compared to enumeration. For example a GP of population 500 and 150 generations that replaces half the population at every generation has to calculate only 500*150/2 = 37500 trees.

References

[1] Holland, J.H. (1975) "Adaptation in natural and artificial systems", Ann Arbor, MI: The University of Michigan Press. 2nd edition. (1992)

[2] Syswerada, G. (1991). Schedule Optimization using Genetic Algorithms" in Davis (ed.), Handbook of Genetic Algorithms, 332-349

[3] J.R. Koza, "Hierarchical genetic algorithms operating on populations of computer programs " Proceedings of the Eleventh International Joint Conference on Artificial Intelligence IJCAI-89, Morgan-Kauffman, 768-774.

[4] Andre, David, Bennet III, Forrest H, and Koza, John R. 1996. Discovery by genetic programming of a cellular automata rule that is better than any known rule for the majority classification problem. Genetic Programming 1996: Proceedings of the First Annual Conference, July 28-31, 1996, MIT Press

[5] Howley, Brian. 1996. Genetic Programming of near-minimum-time spacecraft attitude maneuvers. Genetic Programming 1996: Proceedings of the First Annual Conference, July 28-31, 1996, MIT Press

[6] S.Calderoni and P. Marcenac. Genetic Programming for automatic design of self-adaptive robots. Genetic Programming: Proceedings of the First European Workshop. Springer, Paris, France 1998

[7] John R. Koza, Evolution And Co-Evolution Of Computer Programs To Control Independently-Acting Agents.. From Animals to Animals: Proceedings of the First International Conference on Simulation of Adaptive Behavior, 24-28, September 1990

[8] Futuyma, D.J. & Slatkin, M. 1983. Coevolution. Sunderland, MA: Sainauer

[9] Jansen, D.H. 1980. When is it coevolution? Evolution 34:611-612.

[10] Peter J. Angeline, Jordon B. Pollack. Competitive Environments Evolve Better Solutions for Complex Tasks (1993) Proceedings of the 5th International Conference on Genetic Algorithms (GA-93)

[11] S.A. Kaufmann and S. Johnsen, Co-Evolution to the Edge of Chaos: Coupled Fitness Landscapes, Poised States, and Co-Evolutionary Avalanches, in: C.G. Langton, C. Taylor, J.D. Farmer and S. Rasmussen, Artificial Life II: Proceedings of the Second Artificial Life Workshop, (Addison-Wesley, 1992), 325-369.

[12] Hillis, W. D. (1992). Co-evolving parasites improve simulated evolution as an optimization procedure. In Langton, C. et al. (Eds.), Artificial Life II. Addison Wesley.

[13] Holland, J. H. (1985). Properties of the bucket brigade. In Proceedings of an International Conference on Genetic Algorithms. Hillsdale, NJ. Journal of Evolutionary Economics 7:219-254

[14] Mitchell A. Potter, Kenneth A. De Jong, Cooperative Coevolutionary Approach to Function Optimization (1994). Parallel Problem Solving from Nature -- PPSN III

[15] Axelrod, R. 1987. The evolution of strategies in the iterated prisoner's dilemma. In L.D. Davis, (Ed].), Genetic Algorithms and Simulated Annealing, 32-41. New York, Morgan Kaufmann.

[16] Cliff, D. & G. F. Miller, G.F. 1995. Tracking the Red Queen: Measurements of adaptive progress in coevolutionary simulations". In Moran, F., Moreno, A.,Merelo, J.J. & Cachon, P. (Eds.) Advances in Artificial Life: Proceedings of the Third

European Conference on Artificial Life (ECAL95). Lecture Notes in Artificial Intelligence 929: 200-218. Springer Verlag.

[17] Price, T.C. 1997. Using co-evolutionary programming to simulate strategic behavior in markets. Journal of Evolutionary Economics 7:219-254.

[18] Phelps, S., Parsons, S., McBurney, P., & Sklar, 2002. Co-evolution of Auction Mechanisms and Trading Strategies: Towards a Novel Approach to Microeconomic Design. In Proceedings of ECOMAS-2002 Workshop on Evolutionary Computation in Multi-Agent Systems, at Genetic and Evolutionary Computation Conference (GECCO-2002).

[19] Allen, F. & Karjalainen, R. 1999. Using genetic algorithms to find technical trading rules. Journal of Financial Economics 51:245-271.

[20] Neely, C., Weller, P., & Dittmar, R. 1997. Is Technical Analysis in the Foreign Exchange Market Profitable? A Genetic Programming Approach. Journal of Financial and Quantitative Analysis 32:405-26.

[21] Pazzani, M., Mani, S., & Shankle, W.R. 1997.Beyond concise and colorful: Learning Intelligible Rules. In Proceedings of the Fourth International Conference on Knowledge Discovery and Data Mining, 235-238. Newport Beach, CA: AAAI Press.

[22] Miller, G.A. 1956. The magic number seven, plus or minus two: Some limits on our capacity for processing information. Psychological Review 63:81-97.

[23] Laird, J.E., P.S. Rosenbloom, A. Newell.1986. Chunking in Soar: The Anatomy of a General Learning Mechanism. Machine Learning 1:11-46.

[24] Quinlan J.R. 1987. Simplifying decision trees. International Journal of Man-Machine Studies 27:221-334.

[25] Shepherd, B. An appraisal of a decision-tree approach to image classification. In Proceedings of the Eight International Joint Conference on Artifical Intelligence, 496-501. Philadelphia , PA: Morgan Kaufman.

[26] Buntine, W. & Nibblet, T. 1992. A further comparison of splitting rules for decision-tree induction. Machine Learning 8:75-86.

[27] Clark, P. & Nibblet, T. 1989. The CN2 Induction Algorithm, Machine Learning 3:261-283.

[28] Mingers,J. 1989. An empirical comparison of pruning methods for decision tree induction.Machine Learning 4:227-443.

[29] A. Blumer,A., Ehrenfeucht, A., Haussler, D., & M.K.Warmuth, M.K. 1987. Occam's Razor.Information Processing Letters 24:377-380.

[30] Rissanen, J. 1978. Modeling by shortest data description. Automatica 14:465-471.

[31] Domingos, P. 1999. The Role of Occam's Razor in Knowledge Discovery. Data Mining and Knowledge Discovery 3:409-425, 1999.

[32] Jensen, D. & Cohen, P.R. 2000. Multiple Comparisons in Induction Algorithms. Machine Learning, 38:309-338.

[33] Stephen W. Bigalow, John Wiley & Sons; 1 edition (December 21, 2001) Profitable Candlestick Trading: Pinpointing Market Opportunities to Maximize Profits.

[34] Alfred Cowles 3rd,"Can Stock Market Forecasters Forecast", Econometrica, Vol 1, Issue 3, July 1933. 309-324

[35] Brown, S., Goetzmann W. and Kumar A. (1998). 'The Dow Theory: William Peter Hamilton's Track Record Reconsidered', Journal of Finance, 53(4): 1311-1333.

[36] Eugene F. Fama, The Behavior of Stock-Market Prices, Journal of Business, Volume 38, Issue 1 (January 1965), 34-105

[37] Charles H. Dennis, Comment: The information Content of daily market indicators, Journal of Financial and Quantitative Analysis, Volume 8, Issue 2(Mar 1973) 193-194

[38] Jack Treynor, Robert Ferguson, "In defense of technical analysis", Journal of Finance, Volume 40, Issue 3, Papers and Proceedings of the Forty Third Annual Meeting American Finance Association, Dallas, Texas, December 28-30, 1984 (July 1985), 757-773

[39] Eric H Sorensen, In defense of technical analysis: Discussion, Journal of Finance, Volume 40, Issue 3, Papers and Proceedings of the Forty-Third Annual Meeting American Finance Association, Dallas, Texas, December 28-30, 1984 (July 1985), 773-775

[40] David P. Brown, Robert H. Jennings, On Technical Analysis, Review of Financial Studies, Volume 2, Issue 4 (1989), 527-551

[41] Jeffrey A. Frankel, Kenneth A. Froot, Chartists, Fundamentalists, and trading in the Foreign exchange market, The American Economic Review, Volume 80, Issue 2, Papers and Proceedings of the Hundred and Second Annual Meeting of the American Economic Association (May, 1990), 181-185

[42] Salih N. Neftci, Naive Trading Rules in Financial Markets and Wieneer-Komogorov Prediction Theory: A Study of Technical Analysis, Journal of Business, Volume 63, Issue 4 (Oct. 1991), 549-571.

[43] William N. Goetzmann, Patterns in Three Centuries of Stock Market Prices, Journal of Business, Volume 66, Issue 2 (Apr., 1993), 249-270

[44] Bojarczuk, C.C., Lopes, H.S., & AA Freitas, Data Mining with Constrained-syntax Genetic Programming: Applications in Medical Data Sets. In Proc Intelligent Data Analysis in Medicine and Pharmacology - a workshop at MedInfo-2001, London, September 2001.

[45] http://lancet.mit.edu/ga/