Internal Report 2010-06 June 2010

Universiteit Leiden
Opleiding Informatica

Using Genetic Programming and Evolution Strategies to Evolve Profitable Stock Market Strategies

Jori van Lier

MASTER'S THESIS

Leiden Institute of Advanced Computer Science (LIACS)
Leiden University
Niels Bohrweg 1
2333 CA Leiden
The Netherlands

Using Genetic Programming and Evolution Strategies to Evolve Profitable Stock Market Strategies

Jori van Lier (jvanlier@me.com)
Supervisors: Prof. Dr. Thomas Bäck (baeck@liacs.nl),
Dr. Michael Emmerich (emmerich@liacs.nl)

Leiden Institute of Advanced Computer Science
Leiden University

June 28, 2010

Abstract

This thesis presents a hybrid Genetic Programming and Evolution Strategies algorithm, applied to optimizing stock market strategies. The Genetic Programming algorithm uses a novel quad trading rule architecture, which is cooperatively co-evolved. The Evolution Strategies algorithm is used to fine-tuning the constants in the trading rules. The trading rules are trained for a combined long and short market strategy, and therefore the performance has low correlation with the standard buy-and-hold benchmark. Experimental results are presented and are mostly favorable, even with transaction costs taken into account.

Earlier research mostly tested performance by using just a single training period and a single validation period. Using this simple method may yield reasonable performance, but it does not ensure a positive real-life performance. Therefore, a novel trading simulation has been presented in this paper. A moving training/validation window is used, and the Genetic Programming algorithm is constantly restarted to find better rules. The best rule is used for trading at each trade interval. By doing this simulation, the trading rules are tested on highly diverse data, leading to a much more comprehensive test than simply testing on a static validation set. Unfortunately, performance from static experiments does not translate one-to-one to the dynamic experiments with the simulation, and performance is mostly worse with this more accurate testing method. Still, for some circumstances, it is possible to beat the buy-and-hold benchmark.

Contents

1 Introduction

The financial markets and the vast amounts of data generated by them have been the subject of research for many years. The reasons are simple. It has always been a challenge for computer scientists to find order in chaos and the potential rewards in this case may be substantial.

The groundwork for these prediction models starts with analysis of the market and fortunately there is a lot of existing literature and theory on this field. A global overview will be presented in Section 1.1, among with the two most commonly used market analysis disciplines: fundamental analysis and technical analysis. Section 1.4 introduces the concept of trading rules and Sections 1.2 and 1.3 explain the two Evolutionary Algorithms used. Section 2 will discuss existing published work related to this research. Section 3 explains the model in detail and Section 4 presents the results of the experiments with this model. The practical value of the experimental results are discussed in Section 5. Finally, the conclusions are presented in Section 6 and avenues for future research are highlighted in Section 7.

1.1 Market Analysis

Investors and researchers alike are highly divided on whether market analysis with the goal of forecasting has any practical value. One of the cornerstones of modern finance, the Efficient Market Hypothesis (EMH), effectively rules out the possibility of forecasting financial time series in it's crudest form. An early version of this hypothesis has been around since the early 20^{th} century, which was later empirically confirmed to be correct in the early 1960s [7], 1970s [8] and many times since. The logic behind the hypothesis is based on the finding that if it were possible to forecast returns, many investors would use it to generate unlimited profits, a situation which cannot be maintained in a stable economy.

The EMH is associated with the idea of a "random walk", which is a term in finance loosely used to characterize a price series where all subsequent price changes represent random departures from previous prices [16]. According to the hypothesis, financial assets already reflect all known information and instantly change to reflect any new information that might become available to the public domain. Therefore, it is impossible to consistently outperform the market without insider information or an unlikely amount of luck.

Since the inception of the EMH, researchers have been seeking methods to disprove the EMH, with varying degrees of success. By the start of the 21st century, many financial economists and statisticians began to believe that stock prices are at least partially predictable and claimed that these predictable patterns enable investors to earn excess risk adjusted rates of return [16] .

A recent discussion on the EMH concerning forecasting [26] states that innovative new forecasting models may have a certain "window" of opportunity until the forecasting method self-destructs because too many market participants start using it, causing the information to be absorbed into the price data. Following up on this, the authors stated that prediction models in published research may appear to be successful, but would not be if it was applied in the present. The reasoning is that in almost all cases, a model is

backtested on historic data, in a time that the model was not known and market behavior was different. Would the model be known and exploited by a large amount of investors, the behavior of the market would likely have been altered in such a way that the model's expected profits would not be achievable. Ultimately, however, the authors of [26] do conclude that the first users of a novel prediction method may experience short-lived gains, until the method is widely adopted or discovered by other researchers. A large amount of different prediction models has emerged and will continue to emerge from this constant innovation race.

We can distinguish various disciplines of market analysis that, when used correctly, should increase the odds of an investor making a profitable decision. The two most common analysis methods will be outlined in the following sections.

1.1.1 Fundamental Analysis

Fundamental analysis attempts to value a stock based on past and present data about company's financial statements and health. It involves going over balance sheets to find information on facts such as earnings, costs, cash flow, etc. This type of analysis also requires insight into the market sector and comparisons with rival companies are needed in order to get a good picture on a company's health. A common strategy is to buy relatively low price to earning ratio stocks compared to others in the same business sector, with the assumption that over time the stock should rise in price if all other conditions remain the same.

Benjamin Graham [12] laid the groundwork for present-day fundamental analysis in 1934. This type of market analysis is still used a lot today, but not as extensively as technical analysis, which is discussed in the next section. Also, since fundamental data mostly relies on yearly financial statements and quarterly reports, fundamental data tends to lag behind the price data. This property makes this kind of data less than ideal for usage in a computerized prediction method.

1.1.2 Technical Analysis

Supporters of technical analysis claim that the information obtained by fundamental analysis is already reflected in the stock price, essentially making fundamental analysis pointless. Instead they rely on technical indicators and recurring chart patterns as forecasting methods. The following definition for technical analysis was coined by J. Murphy: "Technical analysis is the study of market action, primarily through the use of charts, for the purpose of forecasting future price trends" [18] . The term "market action" refers to the price and volume information freely available to everyone.

There are many facets to technical analysis. Recurring chart patterns can be used for forecasting the upcoming trend. Work on these patterns started through publications by Charles Dow in the late 19th and early 20th century. After his death, his theories were bundled as the "Dow Theory" in 1932. Work continued with techniques on identifying trends[1], resistance- and support levels.

[1]Market trends: the terms bull market and bear market describe upward and downward movements respectively.

More recently, as computing power became more easily available, numeric technical indicators and oscillators have seen more and more usage. Indicators are calculations based on the publicly available market data that measure things such as the current trend, volatility and momentum. There are two main categories of indicators: leading and lagging. Leading indicators are predictive tools that precede price movements. Lagging indicators serve as a method to confirm a direction since it follows the price data. A special category of indicators are the oscillators, which range between two predetermined bounds, such as 0 and 100.

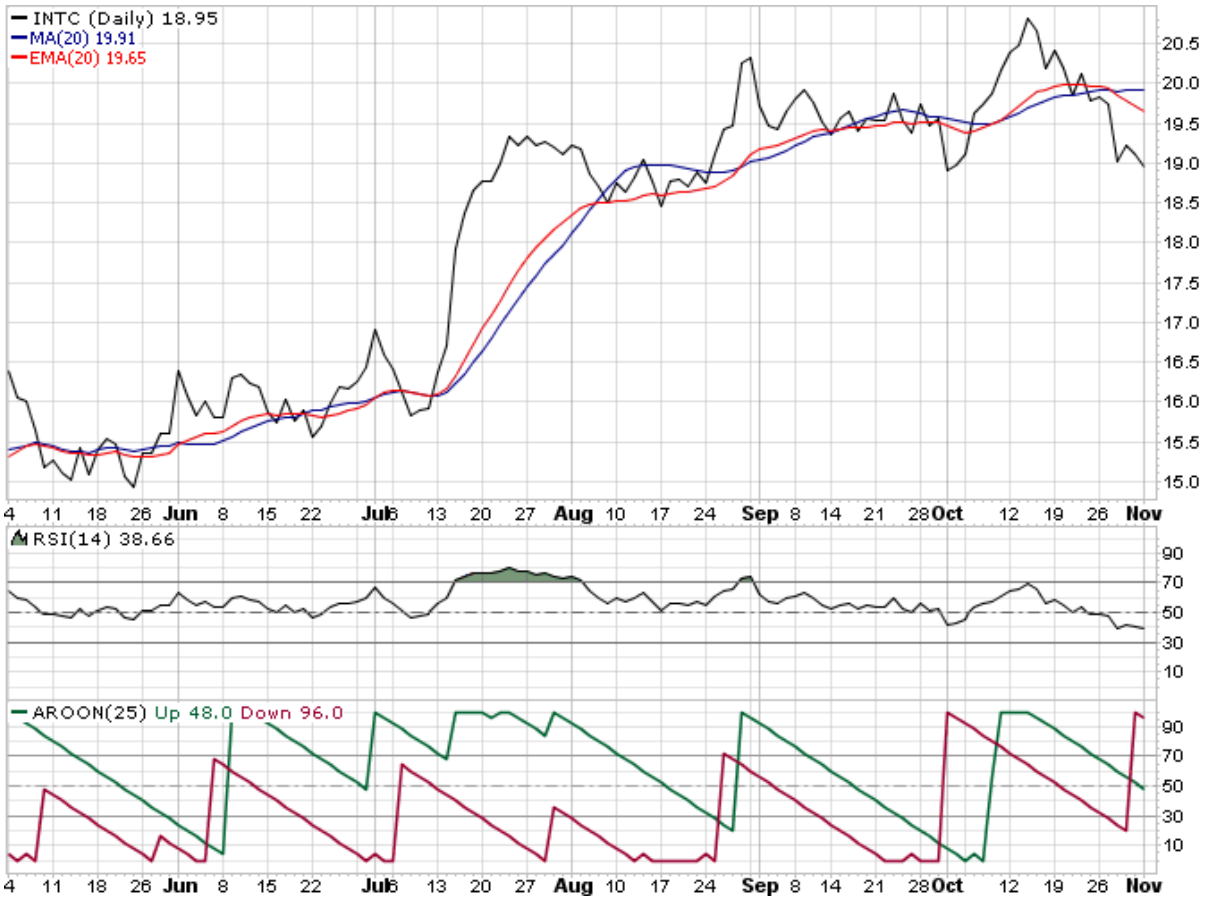

Figure 1: Six months of stock price history for Intel Corp, with 20-day SMA (blue), 20-day EMA (red), 14-day RSI and 25-day Aroon indicators. Image source: StockCharts.com.

The simplest example of an indicator is the moving average, used to smooth out the erratic day-to-day market data. The simple moving average (SMA) is calculated as the average of the last N days:

$$SMA = \frac{p_d + p_{d-1} + \ldots + p_{d-M}}{N}$$

where M is $N-1$ and the prices for day d to day $d-M$ are given by p_d, p_{d-1}, …, p_{d-M}. The exponential moving average (EMA) is a weighted average which values recent data

more strongly than old data, defined as:

$$EMA_d = EMA_{d-1} + \alpha \times (p_d - EMA_{d-1})$$

with the smoothing factor α:

$$\alpha = \frac{2}{N+1}$$

The EMA, in theory, uses an infinite amount of past data. Therefore, it is necessary to either go back far enough, or use a SMA for the initial values in an EMA calculation.

The moving average is a typical lagging indicator. Some traders combine a long and a short moving average, where the crossing of the two averages may indicate a change in trend. Moving averages may also be used to smooth out intraday data (e.g. by using hourly periods), weekly data or even monthly data. See Figure 1 for an example of the two moving averages, in which the EMA can be clearly seen to react more strongly to recent changes than the SMA

Another popular indicator is the Relative Strength Index (RSI), which shows how strongly a stock is moving in its current direction by comparing the magnitude of recent gains to recent losses. The indicator is plotted between 0 and 100. Any value above 70 indicates that the stock is "overbought", while values below 30 indicate that it is "oversold". These situations suggest that the price has been unreasonably pushed to these levels and that the trend may reverse soon. Consider the period in July in Figure 1. The RSI signals an overbought situation quite some time before the price actually falls.

The formula to calculate the RSI is defined as:

$$RSI = 100 - \frac{100}{1 + RS}$$

where RS is the Average Gain divided by the Average Loss over the last N periods. The default amount of periods is 14.

The Aroon indicator can be used to determine whether a stock is oscillating or trending and the likelihood that the trend will reverse. It is made up of two lines plotted between 0 and 100 called Aroon Up and Aroon Down, which measure the strength of the uptrend and the downtrend respectively. The lower the Aroon Up, the weaker the uptrend and the stronger the downtrend and vice-versa. The indicator looks at how recent the latest highs and lows have appeared while determining the value of the two lines. The exact formulas for Aroon Up and Aroon Down are defined as:

$$Aroon(up) = \frac{N - [\texttt{num periods since highest close}]}{N} \times 100$$

$$Aroon(down) = \frac{N - [\texttt{num periods since lowest close}]}{N} \times 100$$

See Figure 1 for an example. The Aroon indicator shows a strong uptrend from early June until mid August.

There are many more technical indicators and the three indicators outlined above is

just a small selection. All technical analysis indicators and oscillators are subject to human interpretation. There are always situations where indicators give conflicting signals or are simply unusable for a certain market condition. The challenge lies in finding an optimal combination of technical indicators for a given market condition and drawing the correct conclusions from them.

1.2 Genetic Programming

Genetic Programming [15, 2] is a development in the field of Evolutionary Algorithms, where individuals in the population are represented by programs in the form of trees. The Genetic Program (GP) evolves these programs into new programs using mutation and crossover operators, selecting individuals with high fitness values for reproduction. This process stops when a predetermined amount of iterations is reached or when no improvement has been found in a number of previous generations.

The evolution of programs can be summarized as follows:

1. *Initialization*: Create an initial random population of n programs and evaluate the fitness of each program using the objective function.
2. *Selection*: Select n programs for reproduction. The selection operator is probabilistically biased towards individuals with higher fitness.
3. *Crossover*: Recombine between individuals.
4. *Mutation*: Mutate individuals.
5. *Evaluation*: Evaluate the fitness of the modified individuals.
6. Set the current population to the new offspring population.
7. Repeat steps 2-7 until the stopping condition is met.

1.3 Evolution Strategies

Evolution Strategies (ES) [2] is a special type of Evolutionary Algorithm, which, in contrast with Genetic Algorithms, allows self-adaption of strategy parameters. The emphasis is on complex mutations, but there are various crossover operations as well. Deterministic selection is used, either in a $(\mu+\lambda)$ form, which only accepts the same quality or improvements, or (μ, λ) in which deterioration is possible. The parent population is described by μ, the offspring is λ, and $\lambda > \mu$ must hold. An ES generates an offspring surplus in order to enable self-adaption.

In its complex form, the ES uses a covariance matrix for mutations, and strategy parameters (step sizes, rotation angles) are part of an individual. These are tuned in addition to the objective variables.

Crossover operators for recombination are: *discrete recombination* and *intermediate recombination*. Discrete recombination randomly selects a variable from either parent one or parent two to create a new individual. For intermediate recombination, arithmetic

means are calculated for each variable. These operators also exist in global forms, which consider all parents in the parent population.

Termination is typically after a given number of generations, function evaluations, or some improvement measure.

The ES evolution process is as follows:

1. *Initialization*: The objective variables and the angles for covariance matrix mutation are typically initialized at random. The step sizes are often initialized using a certain formula that incorporates a rough measure of the distance to the optimum.

2. *Evaluation*: Evaluate the fitness of the initialized individuals.

3. *Recombination*: Iteratively generate λ offspring from the parent population (only for $\mu > 1$).

4. *Mutation*: Normally distributed variations, applied to all individuals.

5. *Evaluation*: Evaluate the fitness of the modified individuals.

6. *Selection*: Deterministically select μ individuals with the highest fitness. Include parents for $(\mu + \lambda)$ selection, only consider the offspring for (μ, λ) selection.

7. Repeat steps 3-6 until the stopping condition is met.

1.4 Trading Rules

This section provides a general introduction to trading rules. Multiple options for evaluation are explained and some examples for a possible grammar are given.

Trading rules are represented by decision trees, recursively constructed from a predefined set of functions and terminals. The GP can be used to evolve these decision trees into new decision trees using mutation and crossover operators. The best trading rules are probabilistically selected according to a fitness function and used to produce the next generation. Eventually, the stopping condition for the GP is reached and the single best trading rule – the rule with the optimal fitness value – could be used to make trade decisions.

A possible function set includes arithmetic operators such as $+, -, \times$ and boolean operators such as `AND, OR, NOT`. Terminals are often comprised of real constants and real functions that return a single value, such as price or volume data.

There are two ways to determine the action corresponding to the evaluation of a decision tree. One method is by using `if-then-else` constructs and have the actions (e.g. buy, sell) as nodes in the tree. The second method uses boolean evaluation for the entire tree, and the action is hard coded into the program. A `true` evaluation could result in a buy action, and a `false` evaluation could result in a sell action. This method is used in this research.

Variations on the standard single trading rule system also exist. Cooperative coevolution with two trading rules can be used to have separate rules for buying and selling, or one for entering and exiting long positions and entering and exiting short positions.

An example of a trading rule is presented in Figure 2. This trading rule can be represented in infix notation: `(vol > 500) AND (MA > 23.3)`. This rule evaluates to true if the volume for the current day is higher than 500 and the moving average is higher than 23.3. The corresponding action is determined by the model.

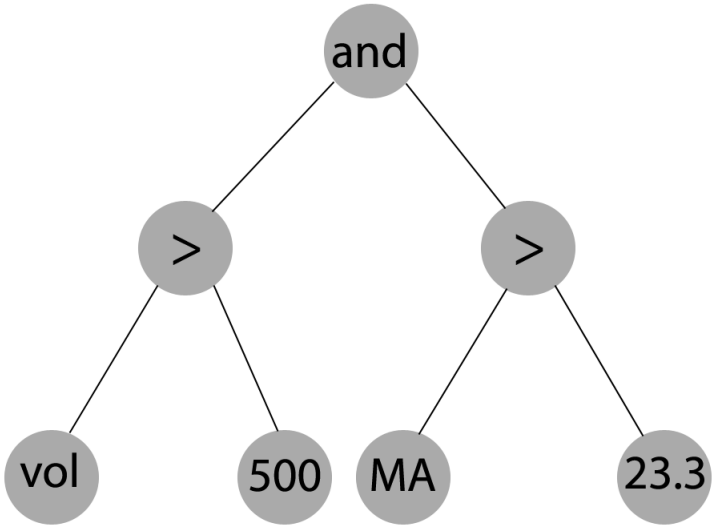

Figure 2: Example of a trading rule. The "vol" node is for volume information and the "MA" node is for a moving average. Boolean evaluation of the tree results in either `true` or `false`, which could mean, "buy" or "sell".

2 Literature Overview

In Section 1.1 we discussed work on prediction methods achieved with classical (non-evolutionary computation) methods. Next we will review results within the evolutionary computation field.

The emphasis of this paper is on is generating and optimizing trading rules using GP. This is not the only application of GP algorithms to computational finance. Price prediction has also been attempted [23, 11] by evolving equations to calculate absolute price values, based on historic price data. More recently, portfolio optimization algorithms have started to emerge [14, 5, 29, 30, 28]. These other applications are beyond the scope of this paper, but serve to show how versatile GPs can be. The interested reader could follow the references to learn more.

The trading rule approach does not involve any actual price prediction, instead focusing on evolving an optimal strategy for a single series of price data. Out of all natural computing methods, this application appears to be best suited to GPs, since it is the most flexible method to generate trading rules modeled as decision trees. There are also examples of successful Genetic Algorithm-based trading rule optimization algorithms, but these methods are more restricted than GP methods. The size of trading rules are often fixed and there is less flexibility in the structure.

Section 2.1 provides a global overview of existing work and sketches a basic time

line with changes to GP architectures and improvements. Finally, Section 2.2 outlines a number of interesting recent research papers.

2.1 Global Overview of Existing Work

The early research on technical analysis may provide additional insight in the history of trading rule-based trading. The first research in the 1970s and 1980s on trading rules and their profitability supported the Efficient Market Hypothesis [9, 8]. This means that it is not possible to gain consistent profits using these ex-ante trading rules. Recent work seems to suggest otherwise. According to Summers et al [25], trading rules derived entirely from a particular time period may have validity over a later time period, and can even surpass the predictive power of rules derived from more recent data.

With the emergence of new technologies and applications, in particular evolutionary algorithms, it is now possible to generate these rules automatically. Genetic Algorithms was the first approach attempted in financial applications, but for generating trading rules, the flexible Genetic Programming approach is a more appealing option. The first attempts on the stock exchange markets did not show any excess returns over the buy-and-hold[2] approach [6, 1], but recent work has been more encouraging [19, 10].

2.2 A Detailed Look at Existing Work

Becker and Seshadri (2003) [4] developed a GP system that outperforms buy-and-hold, even while taking transaction costs into account. They included a number of special features: a complexity-penalizing factor, a fitness function that considers consistency of performance, and a coevolution of a separate buy and sell rule. The fitness function is calculated as the number of periods (of 12 months) with well-performing returns, modified by the complexity-penalizing factor. Consistency of performance was implemented by comparing the returns to buy-and-hold returns and returns on risk-free interest rate.

Potvin et al (2004) [20] consider stocks offered by individual companies, rather than market indices, as most do. The function set includes classical arithmetic, boolean and relational operators, plus boolean if-then-else and a number of real constants. Real functions from technical analysis include: RSI, and rate of change (ROC). Terminals are numeric and boolean constants, plus constants and variables related to stock prices and volumes. Positions are adjusted daily using true (buy) and false (sell) signals. The raw fitness of a trading rule is its excess return over the buy-and-hold approach, evaluated on historical data. Results are positive when tested on the training data, but slightly negative when tested on out-of-sample data. More analysis points out that the trading rules are generally beneficial when the market falls or when it is stable. Buy-and-hold is a better option if the market is steadily rising.

An alternative example of evolving decision trees is provided by Zhang (2005) [31]. The method explored by Zhang is a combined GP and Evolution Strategies (ES) approach,

[2]Buy-and-hold: the standard benchmark approach. A passive strategy which entails buying stocks with the intention of holding them indefinitely. An active trading strategy is often compared with the buy-and-hold strategy on a basket of the same stocks.

similar to the approach described in this thesis. But unlike this thesis, Zhang's algorithm marks each day as "buy" or "do not buy" according to a decision tree. Selling is always done 50 days after buying and if a day is marked as "buy", a 4% profit should be possible within that timeframe. The ES is used after each GP iteration to optimize the value of numeric constants in the x best trees. Sell rules and buy rules were evolved as pairs. Out of the market, only the buy rule is considered and in the market, only the sell rule is considered. The rules are considered to be separate species: buy rules only crossover with other buy rules, sell rules only crossover with other sell rules.

Unlike this thesis and the other works in this section, Zhang went with a classification approach by marking each day according to a decision tree. A fixed time frame is used in combination with a clear goal for that timeframe. Equities are always kept in possession for 50 days. In contrast, this thesis focuses on evolving the most profitable active trading strategy for an unlimited timeframe. However, the hybrid GP and ES approach served as a source of inspiration for this thesis.

Mallick et al (2008) [17] seek to empirically test GP generated trading rules on the component stocks of the Dow Jones IA to obtain evidence that technical analysis rules can yield superior profit. Brokerage provision is incorporated into the fitness function. The results support the hypothesis that trading rules constructed by GP outperform buy-and-hold for the chosen time period when the market is falling. In case of rising markets, the performance compared to buy-and-hold cannot be accurately predicted.

Saks and Maringer (2008) [22] employ GP to discover statistical arbitrage[3] strategies on the banking sector in the Euro Stoxx universe. A preliminary statistical analysis reveals high correlation within stocks in the same business sectors.

Arbitrage portfolio's are constructed, which means that the proceedings from short selling[4] stocks are used to initiate long positions in other stocks. This is a self-financing scheme, and the profits made should be in excess of the risk-free rate and virtually uncorrelated with the market index. Another interesting thing to note about this research is that the authors used intraday data (sampled on an hourly basis), instead of the more commonly used closing prices. The data comprises volume-weighted average prices and volume, covering the time period from 01-Apr-2003 to 29-Jun-2007, for a total of 30 banks in the Euro Stoxx 600 index. The data is heavily preprocessed to remove intraday biases and incorporate volatility information. A portfolio cannot be constructed if there is no opportunity for short selling, a situation that can occur if the trading rules take a bullish view across the board. When forecasts facilitate portfolio construction, this is done on a volatility adjusted basis. Two methods are used for decision tree construction. The first uses a single tree structure, where false means short and true means long. The second method considers a dual tree structure in conjunction with cooperative co-evolution. For this method, program evaluation is contingent on the current market position, i.e., the first tree indicates the long entry, while the second enters a short position. A fairly re-

[3]Statistical arbitrage: a concept where self-funding portfolios are sought where one can expect non-negative pay-outs at any point in time. Negative pay-outs are accepted with a small probability, which ideally converges to zero [22].

[4]Short selling: A method which involves selling a stock without actually having it, with the intention of buying it back later at a lower price. Drops in price result in a profit for the investor.

stricted grammar is used to reduce the risk of overfitting and enhance interpretability of the evolved rules. Both methods perform extremely well on out-of-sample data, and the annualized returns are 23.3% and 19.9%.

In a more recent work by Saks and Maringer (2009) [21], trading strategies are evolved on high-frequency tick data of the USD/EUR exchange rate. The paper proposes a quad tree structure, where each tree has a specific task, corresponding to entering and exiting long and short positions. The grammar of the entry rules is a subset of the grammar of the exit rules, which add functions to capture real-life techniques often used by traders such as stoplosses[5], profit targets and durations of trades. Since the rest of the paper is geared towards exchange market trading and the investigation of money management, it is not within the scope of this literature research. The interested reader is referred to [21].

3 The Model

This chapter explains the hybrid GP and ES model in detail. Data preprocessing for technical indicators is discussed in Section 3.1. The trading rule structure and a formal definition of the grammar can be found in Section 3.2. More information on the GP implementation is in Section 3.3. The objective function used by the GP and the ES is detailed in Section 3.4 and the ES implementation is discussed in Section 3.5.

Finally, a graphical representation of the workflow of the entire algorithm is presented in Section 3.6.

3.1 Data Preprocessing

The GP system uses unmodified daily close price data and volume data. The RSI, the Aroon indicator, and three exponential moving averages with windows of different lengths are calculated using the closing prices. The lengths of the moving average windows are 5, 10 and 15 days, corresponding to 1, 2 and 3 weeks. All data is scaled into the range from 0 to 1.

The technical indicators RSI, Aroon and moving averages have been previously discussed in the introduction. Please refer to Section 1.1.2 for the definitions of these indicators.

3.2 Trading Rules

3.2.1 Grammar

The grammar for the trading rules consists of two types of function nodes: boolean operators `NOT, AND, OR, XOR` and numeric comparators $<$ and $>$. Boolean operators may only have other boolean operators and numeric comparators as arguments (children). Numeric comparators may only have terminals as arguments. Terminals are composed of the terminal variables price, volume, three moving averages of different lengths, RSI and Aroon indicators, and constants in the range from 0 to 1. Out of the terminal variables,

[5]Stoploss: automatic exit when the current profit is below a certain level.

volume, RSI and Aroon may only be compared with constants. Price and the moving averages may also be compared with each other.

Trading rules are constructed in such a way that the upper part is always built from boolean operators. The numeric comparators make the transition into terminal variables and constants. These restrictions on tree construction are built into the GP and enforced by all construction and mutation functions, in an attempt to not waste time on evolving solutions that do not make semantic sense. See Figure 3 for an example of a trading rule, where the different layers are marked by colors.

A Backus-Naur Form definition is given below. The grammar is in prefix notation. A trading rule starts with <boolean>.

```
<boolean> ::= "NOT(" <function> ")" |
              <bool_binary> "(" <function> "," <function> ")"

<bool_binary> ::= "OR" | "AND" | "XOR"

<function> :== <boolean> | <comparator>

<comparator> ::= <lt_or_gt> "(" <comp_or_term> "," <comp_or_term> ")"

<lt_or_gt> ::= "<" | ">"

<comp_or_term> ::= <comparator> | <variable> | <constant>

<variable> ::= "Aroon" | "RSI" | "Price" | "Volume" |
               "MA-short" | "MA-medium" | "MA-long"

<constant> ::= <real number from 0 to 1>
```

The *closure* requirement [15] of GP grammars states that each function in the function set should be well defined for any number of arguments. This is not the case in this implementation, since the numeric comparators cannot handle boolean operators as arguments. This issue has been overcome by applying the tree structure restrictions, but the closure requirement is still violated.

3.2.2 Sets of Trading Rules as a Single Individual

The trader simulation is implemented to allow short positions – where a profit is made if prices decline – as well as long positions. Therefore, three possible states are possible for the trader: out of the market, in the market – long and in the market – short.

The default configuration of the GP considers four rules as a single individual. This quad rule structure is termed a "trading ruleset". There are rules for "buy long", "sell long", "buy short" and "sell short". Boolean evaluation is used to determine if an action should be taken. There is a certain synergy between these rules: if buy long and sell long both evaluate to true, buying with these conflicting signals does not make sense. Table 1 details how these rules work together. Note that in order to facilitate a state change (e.g.

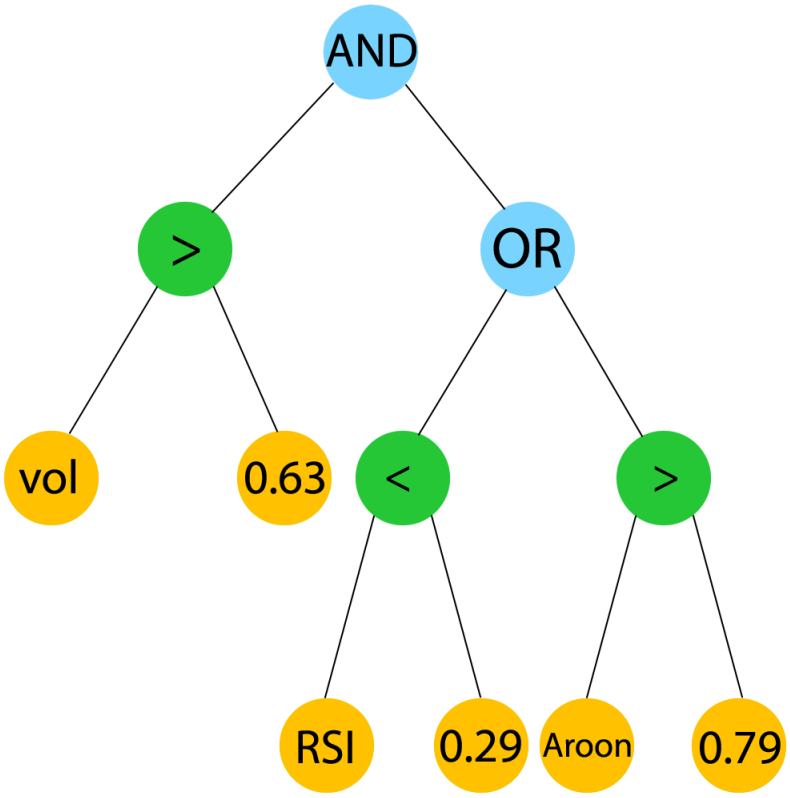

Figure 3: Example trading rule as used in the model. The blue nodes indicate the top layer of the tree: boolean operators. The orange nodes are the terminals, and the green numeric comparator layer makes the transition between the two.

Buy long	Sell long	Buy short	Sell short	Current state	New state
true	false	false	x	Out	Long
true	x	true	false	Out	Short
false	true	false	x	Long	Out
false	true	true	false	Long	Short
false	x	false	true	Short	Out
true	false	false	true	Short	Long

Table 1: Quad rule market transition table. The character x denotes a "don't care" condition.

from out of the market, to in the market – long), at least three out of the four trading rules needs to evaluate to a specific value in order to facilitate a state change.

For comparison purposes, dual rule and single set structures have also been implemented, but the emphasis of this research is on the quad set structure. The dual set uses a rule for "long" and a rule for "short". The table for this set is much simpler, see Table 2. The single rule version is trivial: a true evaluation results in a buying long and a false evaluation results in buying short. With this configuration, the trader is always in the market, and the rule basically forecasts the direction of the price movements.

Long	Short	Current state	New state
true	false	Out	Long
false	true	Out	Short
false	false	Long	Out
false	true	Long	Short
false	false	Short	Out
true	false	Short	Long

Table 2: Dual rule market transition table

3.3 GP Implementation

The GP uses a quad tree structure, which is cooperatively co-evolved. Only trees of the same species may be crossed over with each other.

An elitist system is used to keep a record of the best set. The "best" set is the set with the highest fitness value over all rounds. After completion of a GP run, the best set is the final result. If there is no change to the best set during the last 15 rounds, the GP exits. Otherwise, it continues until the maximum amount of runs (51) is reached.

The population is initialized using the *ramped half-and-half* method [3]. The initial maximum depth is restricted to 5. The first half of the population is initialized using the *full* method: keep generating nodes until the maximum depth is reached. The second half is generated using the *grow* method: switch early to the numeric comparator level with a certain probability (0.2). When the numeric comparator layer has been reached in a certain branch, the next nodes must be terminals.

3.3.1 Genetic Operators

The mutation and crossover operators always use random points to start the operations. Since a specific tree structure has been defined, in which the boolean operators are at the top and numeric comparators make the transition between these boolean operators and the terminals, these structural rules need to be enforced by the crossover and mutation operators. For some operators, instead of selecting some random node as starting point for an operation, we often have to be specific if it should be a boolean operator or a numeric comparator. In some cases this does not matter, and when referring to an "internal node", this is either a non-root boolean operator or a numeric comparator.

Crossover

Trading rules inside a ruleset may only cross over with trading rules of the same species in other rulesets. There are two basic crossover operations. *Subtree crossover* uses two parents and picks a randomly selected internal boolean operator on the first parent. A random child of this node is replaced by a copy of a random internal node of the second parent and its children. This operation only yields one child – the second parent is discarded.

Self-crossover uses a single parent and selects a random boolean operator as a base for the operation. One of the children is randomly selected to be replaced by a copy of a another internal node in the tree.

Note that, in order to prevent too much duplication inside a tree, the subtree crossover is highly preferred with a probability of 0.8.

Mutation
When a ruleset has been selected for mutation, one out of six mutations is randomly chosen with equal probability. The selected mutation is applied to all trading rules in the ruleset. Most of these traditional mutations have been slightly modified in order to apply the restrictions to the tree structure. Node selection is always random, but often with restrictions on the node type.

Point mutation selects a node, and replaces this by another node of the same type.

Permutation mutation swaps the location of the children of a non-terminal node.

Hoist mutation selects an internal boolean operator. This node is now the new root of the trading rule, the rest of the trading rule is discarded.

The *expansion mutation* selects a numeric comparator node, and inserts a new subtree on its location. The root of this new subtree is a boolean operator. The *subtree mutation* is similar, but it selects an internal boolean operator to be replaced by the new subtree.

The *collapse mutation* selects an internal boolean operator, changes its type to a numeric comparator, and adds two terminals as its children.

Since we are working with restrictions with regard to tree structure, it may happen that no suitable node can be found for the selected mutation. If this is the case, the mutation is simply omitted.

Selection
Selection is done using tournament selection with a selection size of 5. A group of 5 individuals is randomly chosen from the population. The individual with the highest fitness value in this group is chosen to reproduce. The reproduction operator is chosen according to probabilities. The new offspring can be an unmodified copy, or the result of a mutation or crossover operation.

3.3.2 Parameter Settings

This GP implementation keeps the standard probability settings for mutation and crossover from Koza [15]. See Table 3 for an overview of all important settings. Some parameter settings may have already been mentioned in earlier sections.

Population size	500
Maximum amount of rounds	51
Maximum amount of rounds without improvement	15
Maximum initial tree depth	5
Tournament size	5
Crossover chance	0.6
Mutation chance	0.05

Table 3: Parameter settings

3.4 Objective Function

The objective function is the annualized return:

$$AnnualizedReturn = \frac{TotalReturn}{years}$$

This is calculated by simulating trading during a consecutive period. The starting amount of money is 5000. After each day, if the trader was in the market, the current amount of money is updated with today's movements. One action is allowed after each day. Which action that is, is determined by evaluating the trading rules in the set. This could result in one transaction (buying, selling), or two (instant position switch from long to short and vice-versa). It is assumed that the transactions can be completed using the last known price, instead of a certain price in the future. To review the possible market states transitions, please refer to Table 1.

Instead of buying a certain number of shares, the assumption is made that the entire amount of available money can be invested.

Transaction costs are accounted for. They are calculated by taking 0,14% of the total amount, with a minimum value of 6 and a maximum value of 45. This cost is immediately subtracted from the total money value after the transaction. The transaction cost formula was taken from one of the more popular brokers in the Netherlands.

3.4.1 Fitness Penalty

A problem inherit to GP is that the average size of trees in the population often increases round after round. This poses several problems. It increases computation time, while not necessarily increasing performance. It also does not help with the interpretability of the trees. And, since the goal is to use the ES to optimize the constants inside the decision trees, the ES will likely perform better if there are less parameters to optimize.

A fitness penalty was introduced in an attempt to counteract this effect. A formula was devised such as to not penalize small violations too much. The penalty increases exponentially if the size of the decision tree increases. The maximum allowed size without penalty is 50. For any tree that is over 50, the penalty per rule is calculated as:

$$2^{1 + \frac{Size - 50}{\alpha}}$$

The control parameter α was set at 16. This value has been empirically found to offer good characteristics with regard to increase of the penalty.

The total penalty applied to the objective function of an individual at any time is the sum of the penalty for each rule in the set.

3.5 ES Optimization

The ES algorithm used for this application is Covariance Matrix Adaption Evolution Strategy (CMA-ES) [13]. The algorithm is released under the GNU general public license and is available at [27].

After each round of GP optimization, the ES is initialized to optimize the constants of the top n% of the population. The original constants are given as input, and the output

is replaced in the tree – even if the performance is worse. This is to avoid getting stuck in local optima. The objective function is the same as the the one used for the GP.

The ES is configured in a (1,10) configuration, meaning that each round produces 10 offspring. The best out of this offspring is selected as parent for the following round. Initial standard deviations are set to 0.1. Box constraints are applied – constants have to remain in the range from 0 to 1. The maximum amount of runs is 1000 and the ES may also quit before that if early stagnation is encountered.

3.6 Algorithm Workflow

See Figure 4 for a graphical representation of the workflow of the algorithm.

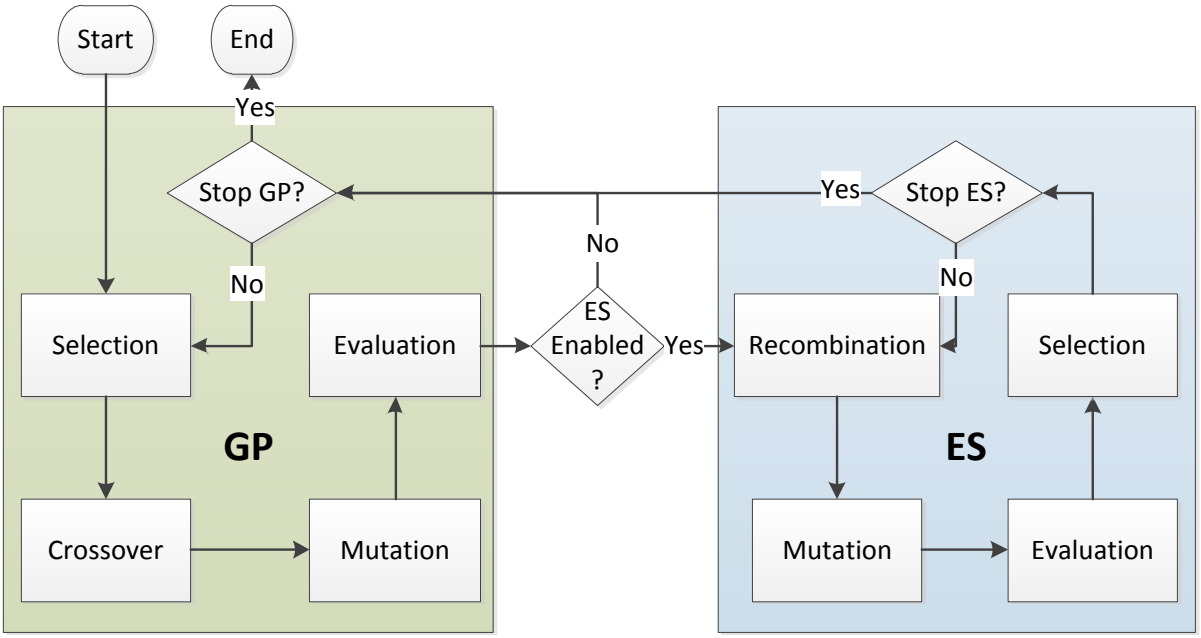

Figure 4: All the GP actions are in the green section on the left. The ES is on the right in the blue section. The ES part is only used if the ES is enabled.

4 Experimental Results

In the following sections, the results of the experiments are shown. Section 4.1 provides information on the three data sets are used throughout the experiments. Section 4.2 tests the GP (without ES) using the single rule, dual rule and quad rule configurations. In Section 4.3, an experiment is attempted where the long and short strategies are separately optimized and evolved, also without the ES integration. These separate strategies are then combined into a single individual to calculate the composite performance. Section 4.4 presents the results of the hybrid GP+ES.

Sections 4.5 and 4.6 attempt to test the GP (with and without ES) by using a more elaborate experiment. The GP is executed multiple times using a moving time window,

basically re-running the GP each day for a duration of multiple years. This provides us with an indication of real-life performance, and the GP is subjected to continually changing training and validation data.

For the experiments in Sections 4.2 to 4.4, good performance is indicated by a high in– and out-of-sample (columns "In" and "Out" respectively) average annualized return, ideally with a low amount of trades. The standard deviation is also shown, along with the two individuals that have the highest in– and out-of-sample annualized return (rows "Highest fitness" and "Highest validation" respectively).

Note that if the "Highest validation" individual also has a near-average or above-average in-sample annualized return, this result can be considered to be especially good. It managed to train well enough to be above-average in-sample, while generalizing extremely well to the validation part of the dataset.

The *Sharpe ratio* [24] is a measure of the excess return per unit of risk. It is calculated using this formula:

$$S = \frac{E[R - R_f]}{\sigma}$$

where R is the asset return, R_f is the return on a benchmark asset such as the risk free rate[6], $E[R - R_f]$ is the expected value of the asset return over the benchmark return, and σ is the standard deviation of the asset excess return: $\sigma = \sqrt{var[R - R_f]}$. The higher the Sharpe ratio, the better the the risk-adjusted performance. For these calculations, a yearly risk free rate of 4% was used. The Sharpe ratio is traditionally calculated using yearly return data. In this case however, monthly data was used to obtain a more accurate reading. In order annualize this value, the result was multiplied by the square root of 12.

4.1 Data and training/validation split

Three datasets are used:

- DJIA: Dow Jones Industrial Average. January 1, 2000 to December 31, 2009 – 10 years.
- SP500: Standard & Poor's weighted index of 500 large-cap US stocks. January 1, 1990 to December 31, 2009 – 20 years.
- TWII: A weighted index from the Taiwan Stock Exchange Corporation. January 1, 2003 to December 31, 2009 – 7 years.

The Dow Jones Industrial Average (DJIA) is a low volatility index of 30 well known large cap US stocks. The S&P 500 (SP500) index consists of 500 large cap US stocks. The Taiwan economy is an emerging market, and therefore the price of the index (TWII) moves with a higher volatility than the other two datasets.

For all experiments, the data is split into 70% training (in-sample) and 30% validation (out-of-sample). Table 4 presents relevant information to these datasets: the amount of time in days in the training and validation periods, and an annualized buy-and-hold

[6]Risk free rate: the interest rate that can be obtained by investing in financial assets that do not have a realistic default risk, e.g. government bonds of developed countries.

Figure 5: Dataset "DJIA": The Dow Jones Industrial Average from January 1, 2000 to December 31, 2009. During this period, the price has decreased by 1069 (-9%).

Figure 6: Dataset "SP500": The S&P 500 from January 1, 1990 to December 31, 2009. During this period, the price has increased by 762 (215%)

Figure 7: Dataset "TWII": The Taiwanese index from January 1, 2003 to December 31, 2009. During this period, the price has increased by 3641 (80%)

benchmark, if one were to buy on the first day and sell on the last day. Figures 5, 6 and 7 display the price movements over time. The vertical green lines give an indication of the training/validation split. The source of these figures is Google Finance. All data has been taken from the Yahoo Finance website and has been checked for consistency.

	Time – years		Time – days		Buy-and-Hold	
	In	Out	In	Out	In	Out
DJIA:	6.8	3.2	1706	809	1.7%	−4.0%
SP500:	14	6	3526	1517	16.2%	0.3%
TWII:	4.8	2	1216	516	17.2%	−2.7%

Table 4: Input dataset statistics: the amount of time in years and days, and an annualized buy-and-hold return, for the in– and out-of-sample periods.

Note that the split may not be exactly 70%-30% due to holidays falling on different days of the week in different years, and being unable to use the first number of days for training since we need starting data for the indicators (such as moving averages). The yearly values have been calculated by dividing the amount of days by 252, a good indication of the average amount of days per year that the market is open in western countries.

4.2 GP: varying setsizes

The results of the following experiments were averaged over 100 runs. The standard deviation is shown, along with the individual that had the highest fitness, and the individual that had the highest validation value.

4.2.1 Setsize 1

Dataset: DJIA

	Annualized Return		Trades		Sharpe Ratio	
	In	Out	In	Out	In	Out
Average:	35.5%	8.1%	162	80	0.94	0.21
Standard deviation:	8.9%	10.6%	52	58	0.14	0.42
Highest validation:	38.4%	**56.6%**	251	171	0.95	1.57
Highest fitness:	**64.2%**	−3.1%	249	72	1.26	−0.44

Dataset: SP500

	Annualized Return		Trades		Sharpe Ratio	
	In	Out	In	Out	In	Out
Average:	73.4%	4.2%	146	171	0.90	0.03
Standard deviation:	19.7%	6.1%	58	102	0.11	0.29
Highest validation:	73.1%	**23.5%**	133	159	0.89	0.73
Highest fitness:	**132.7%**	6.6%	199	211	1.16	0.19

Dataset: TWII

	Annualized Return		Trades		Sharpe Ratio	
	In	Out	In	Out	In	Out
Average:	86.8%	−0.1%	72	37	1.59	−0.07
Standard deviation:	25.5%	8%	37	21	0.28	0.25
Highest validation:	175.3%	**25.7%**	135	53	2.16	0.68
Highest fitness:	**184.5%**	0.4%	129	51	2.15	−0.12

These results show a very high amount of trades, which is most likely due to the fact that the single rule configuration always has to be in the market. A true evaluation means buy long, and a false evaluation means buy short. Still, the average DJIA out-of-sample return is very respectable.

4.2.2 Setsize 2

Dataset: DJIA

	Annualized Return		Trades		Sharpe Ratio	
	In	Out	In	Out	In	Out
Average:	51.2%	2.5%	84	74	1.2	−0.05
Standard deviation:	14.4%	10.1%	45	91	0.22	0.53
Highest validation:	75.8%	**30.9%**	47	41	1.65	1.06
Highest fitness:	**88.3%**	0.5%	137	287	1.49	−0.04

Dataset: SP500

	Annualized Return		Trades		Sharpe Ratio	
	In	Out	In	Out	In	Out
Average:	92.9%	6%	23	31	1.01	0.05
Standard deviation:	20.9%	10.3%	37	51	0.08	0.47
Highest validation:	73.4%	**32.5%**	5	5	0.94	0.94
Highest fitness:	**159.1%**	−1.3%	129	179	1.11	−0.38

Dataset: TWII

	Annualized Return		Trades		Sharpe Ratio	
	In	Out	In	Out	In	Out
Average:	99.7%	-2.6%	29	13	1.7	−0.17
Standard deviation:	28.3%	15.2%	27	16	0.29	0.53
Highest validation:	139.1%	**41.5%**	133	79	2.01	0.82
Highest fitness:	**200.2%**	18.1%	73	21	2.67	0.44

The average out-of-sample results have increased for SP500, but decreased for DJIA and TWII.

4.2.3 Setsize 4

Dataset: DJIA

	Annualized Return		Trades		Sharpe Ratio	
	In	Out	In	Out	In	Out
Average:	39.8%	6%	48	20	1.07	0.02
Standard deviation:	12.7%	14.1%	32	17	0.27	0.69
Highest validation:	51%	**62.2%**	63	27	1.26	1.64
Highest fitness:	**94.4%**	7.6%	51	17	1.82	0.23

Dataset: SP500

	Annualized Return		Trades		Sharpe Ratio	
	In	Out	In	Out	In	Out
Average:	90.5%	9.8%	17	16	1.03	0.21
Standard deviation:	15.7%	11.5%	14	24	0.07	0.42
Highest validation:	96.6%	**39.1%**	41	31	1.06	0.92
Highest fitness:	**140.6%**	−5.2%	33	23	1.21	−0.48

Dataset: TWII

	Annualized Return		Trades		Sharpe Ratio	
	In	Out	In	Out	In	Out
Average:	89.3%	4.2%	21	9	1.67	−0.01
Standard deviation:	23.1%	27.3%	10	7	0.23	0.74
Highest validation:	69.4%	**140.8%**	9	7	1.56	2.15
Highest fitness:	**170.3%**	−2%	25	3	1.95	−0.11

The average out-of-sample results are better for all datasets except DJIA – this is still better for the single rule GP, although the result is reasonable. It beats buy-and-hold by a large margin.

4.2.4 Discussion

It appears that the quad set GP has the best out-of-sample results overall. Clearly the additional complexity obtained through introducing multiple rules helps in capturing the behavior of the time series. The average annualized out-of-sample returns are higher for all datasets, except for DJIA. But the quad set GP has the best "highest validation" value for all datasets.

Two real winner rulesets have been evolved for the DJIA and SP500 datasets using the quad-set GP, since the "Highest validation" rules also have an above-average fitness. These rulesets can be seen in the Appendix.

One interesting thing to note here is that when more rules are used, the amount of trades is drastically reduced. This mostly works in favor for the quad set structure, since transactions costs obviously negatively affect the annualized returns.

In earlier research, the results of the active trading were often compared to the standard benchmark: buy-and-hold. However, since this algorithm can also profit from drops in price, the results show very low correlation with the buy-and-hold performance in Table 4.

The following experiments will only be executed for the quad set GP.

4.3 GP: Long and Short Strategies Separately Evolved

The trading rulesets in the previous experiment were optimized by using a fitness function that does a full trade simulation over a static time period. In theory, this may lead to underdeveloped short-strategies if the data set is mostly favorable for long positions (and vice versa). In this experiment, the optimal long and short strategies are separately evolved. The best rules are then selected and combined into a single individual.

This experiment has only been done with the quad set GP. A total of 100 runs were done per data set.

Dataset: DJIA

	Annualized Return		Trades		Sharpe Ratio	
	In	Out	In	Out	In	Out
Average:	18.3%	3.4%	46	24	0.55	−0.09
Standard deviation:	16.3%	9.5%	36	20	0.56	0.82
Highest validation:	29.9%	**29.2%**	30	26	1.14	1.34
Highest fitness:	**63.4%**	−3.1%	77	7	1.46	−0.36

Dataset: SP500

	Annualized Return		Trades		Sharpe Ratio	
	In	Out	In	Out	In	Out
Average:	34.4%	3%	18	28	0.59	−0.44
Standard deviation:	20.4%	7.2%	22	37	0.42	3.11
Highest validation:	48%	**30%**	5	5	0.8	0.87
Highest fitness:	**98%**	0%	11	2	1.03	−30.69

Dataset: TWII

	Annualized Return		Trades		Sharpe Ratio	
	In	Out	In	Out	In	Out
Average:	34.6%	6.3%	20	9	0.86	0.06
Standard deviation:	24.4%	23%	17	9	0.79	0.68
Highest validation:	30.9%	**107%**	7	3	0.86	1.92
Highest fitness:	**91%**	2.3%	23	15	1.82	0

4.3.1 Discussion

The average and highest returns have decreased using this setup. Surprisingly, the TWII out-of-sample performance increased by 2.1%.

Overall, it appears that if a profitable long strategy and a profitable short strategy are combined, this results in conflicting signals since the two strategies are not completely in sync. This can be explained by looking at the synergy between the four trading rules (see Table 1). For example, buying long is only allowed if "buy long" is true, "sell long" is false, and "buy short" is true. When training for *just* the long strategy, the "buy short" rule is not trained. The problem reveals itself when combining the long and short strategy to form a single individual, since now the "buy short" rule is also used for buy long decisions, even though it not trained for this task.

No improvement can be obtained from this setup, except maybe for emerging markets (TWII), but that cannot be said with any certainty since the in-sample average return is lower, and the best validation is lower as well. Since we are looking for a general increase, this research avenue was dropped from future research in this paper.

4.4 GP and ES

For the following experiments, the ES was integrated into the GP to optimize constants in the decision trees. ES optimization was done after each GP round for the top 1% and top 2% of the population, corresponding to 5 and 10 individuals respectively.

Unfortunately, the ES requires a lot of computational resources. While optimizing 1% of the population with a size of 500, approximately 98% of the execution time is spent on ES optimization. In order to cope with this, the GP parameter that controls how many rounds without a change to the best (elitist) set are allowed before exiting was changed from 15 to 10. All other settings remain the same.

These experiments were only performed with a set size of four, since the previous experiments showed that, overall, this configuration performs better than the two-set and one-set options. A total of 10 runs were done per data set.

4.4.1 Optimizing 1%

Dataset: DJIA

	Annualized Return		Trades		Sharpe Ratio	
	In	Out	In	Out	In	Out
Average:	84%	4.4%	73	29	1.5	−0.01
Standard deviation:	11.5%	14.7%	39	28	0.15	0.8
Highest validation:	65.8%	**36.5%**	45	9	1.29	1.35
Highest fitness:	**106.1%**	−2.5%	73	13	1.66	−0.28

Dataset: SP500

	Annualized Return		Trades		Sharpe Ratio	
	In	Out	In	Out	In	Out
Average:	170.6%	9%	54	24	1.2	0.19
Standard deviation:	67.2%	10.6%	46	32	0.2	0.38
Highest validation:	123.4%	**33.8%**	37	7	0.98	0.91
Highest fitness:	**309.1%**	1.9%	87	5	1.62	−0.01

Dataset: TWII

	Annualized Return		Trades		Sharpe Ratio	
	In	Out	In	Out	In	Out
Average:	160.3%	4.9%	38	14	2.24	0.02
Standard deviation:	35.5%	23.4%	13	9	0.29	0.67
Highest validation:	121.5%	**52%**	27	17	1.88	1.28
Highest fitness:	**209.1%**	−8%	39	9	2.34	−0.14

4.4.2 Optimizing 2%

Dataset: DJIA

	Annualized Return		Trades		Sharpe Ratio	
	In	Out	In	Out	In	Out
Average:	95%	−1.3%	68	28	1.64	−0.24
Standard deviation:	25.2%	5%	24	18	0.27	0.3
Highest validation:	73.6%	**5.2%**	61	31	1.52	0.13
Highest fitness:	**124.2%**	−8.3%	100	38	1.88	−0.59

Dataset: SP500

	Annualized Return		Trades		Sharpe Ratio	
	In	Out	In	Out	In	Out
Average:	225.4%	0.6%	69	54	1.36	−0.21
Standard deviation:	66.8%	6.8%	27	38	0.13	0.42
Highest validation:	289.5%	**17%**	91	57	1.46	0.52
Highest fitness:	**322.9%**	−2.1%	87	65	1.52	−0.26

Dataset: TWII

	Annualized Return		Trades		Sharpe Ratio	
	In	Out	In	Out	In	Out
Average:	190.8%	−5%	39	11	2.21	−0.3
Standard deviation:	40.5%	13.3%	15	6	0.28	0.56
Highest validation:	147.1%	**10.4%**	21	11	1.88	0.25
Highest fitness:	**268%**	−28.2%	55	9	2.47	−1.44

4.4.3 Discussion

As the results clearly show, the ES does an excellent job at finding optimal values for the training period. Unfortunately, this does not go without a hit to the validation period. Fitness values are higher across the board, and validation values are mostly lower. This indicates overfitting. Overly aggressive fine-tuning of the constants is a possible explanation. Consider this branch in a buy long tree: `PRICE` $< a$. The constant a may have been randomly evolved by the GP and received a high fitness value. Now consider figure 8. The

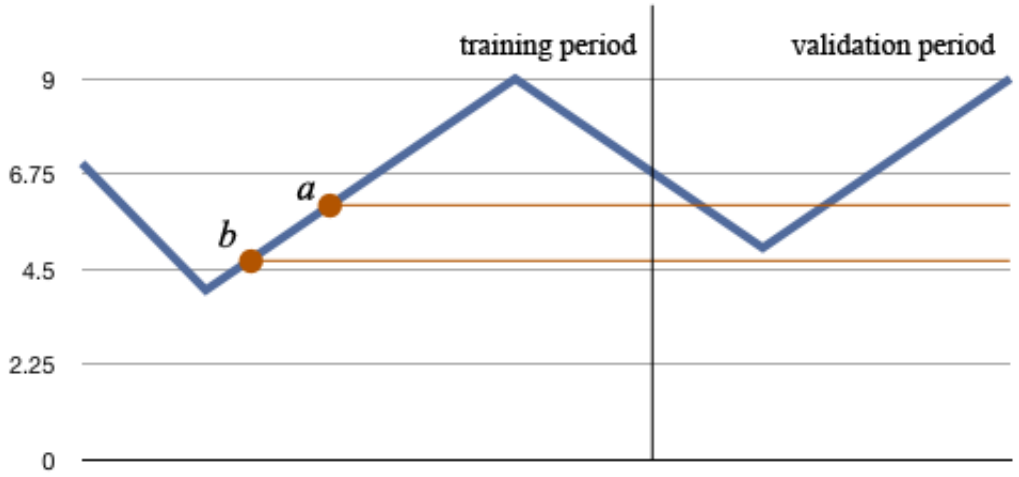

Figure 8: Graphical explanation of overfitting by the ES. Constants are tuned to values that may never be reached again during the validation period.

ES, during its optimization process, may reach a slightly higher fitness value by setting the constant node to the value of b. But unfortunately, the next time the price drops during the validation period, b will not be reached again. However, buying at a would have been highly profitable. The ES fine-tuning results in higher in-sample returns, at the cost of lower out-of-sample returns. The same thing can happen with volume and moving average terminal functions, when compared to constants. The value of the constant may never be reached again.

This effect is even more apparent when the top 2% of the population is optimized, as the results indicate: higher average fitness and lower validation values for all datasets.

In order to make the ES fine-tuning viable, more fuzzy concepts are needed, in lieu of the highly absolute `PRICE < constant` constructs.

4.5 Real-life Simulation: Trading And Retraining Daily (without ES)

In order to simulate a real-life usage of the GP, the following experiment has been proposed: Use a percentage – for example, 70% – of the full dataset for the GP, starting at day 0. This is the training-validation window. This window is shifted 1 day to the right after each successive GP run. The trader simulation starts at the 70% mark. The GP is restarted each day to make today's decision. The set with the highest validation value is from this point on referred to as "best set".

For the initial day, ten GP runs are done, the best set is taken to make the decision for the current day.

For each following day, the window is shifted one day to the right, and one additional GP run is done on this new window, providing a new potential best ruleset. The previous best set will also be re-tested using the current window to provide updated fitness and validation values. The best ruleset out of these two is used to make today's decision.

This process continues until the last day in the dataset is reached. Trading is only

allowed if the best set has a fitness value of at least 20, and a validation value of at least 15. Figure 9 details the workflow of the simulation.

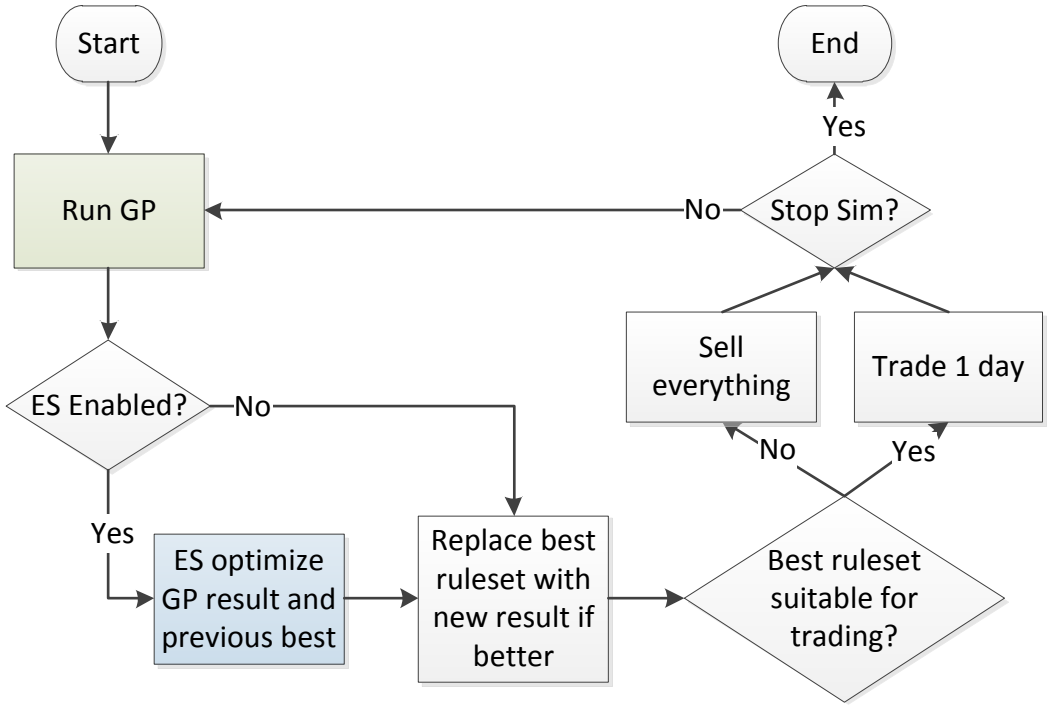

Figure 9: Graphical representation of the simulation workflow. The ES is not used for this experiment, but it is used in the experiment in Section 4.6. The simulation stops when the last day has been reached.

For the DJIA and TWII datasets, the GP training-validation window size is set at 70%, which is about 7 and 5 years respectively. The last 3 and 2 years of the datasets are used for trading. When running the experiments on the 20-year SP500 dataset, 70% proved to be too much, and the performance was not very good. This is probably due to the GP training too much on events that occur very rarely, but have a tremendous impact, such as the market crash (dot-com bubble) in 2001 and the subsequent very fast rise (see Figure 6 in Section 4.1). Thus, for SP500, 35% was used by the GP, also 7 years. A very long time was spent trading though: 13 years.

4.5.1 Results

Dataset	Annualized Return	Sharpe Ratio	Abs. Return	Abs. Return BnH
DJIA	2.1%	−0.38	320	−761
SP500	2.6%	0.31	1726	2444
TWII	−7.2%	−0.13	−819	−865

See Figures 10, 11, 12 for graphs of the the model's cash value over time, compared to buy-and-hold, starting at 5000.

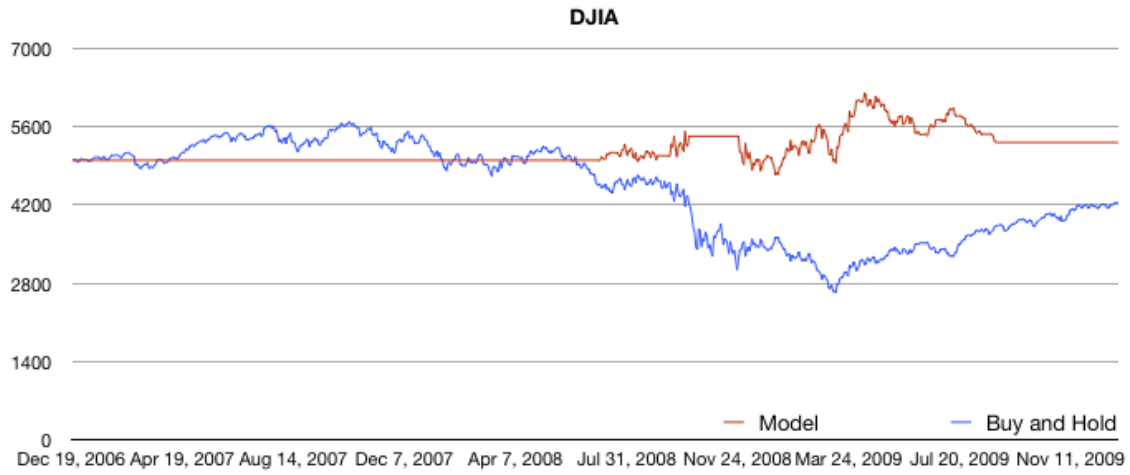

Figure 10: Simulation GP – DJIA: 320 profit over 4 years.

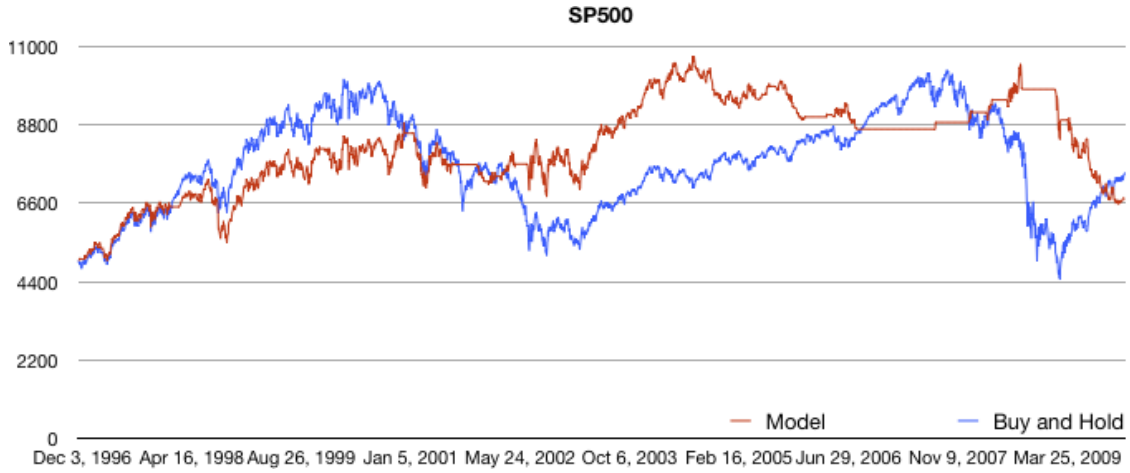

Figure 11: Simulation GP – SP500: 1726 profit over 13 years.

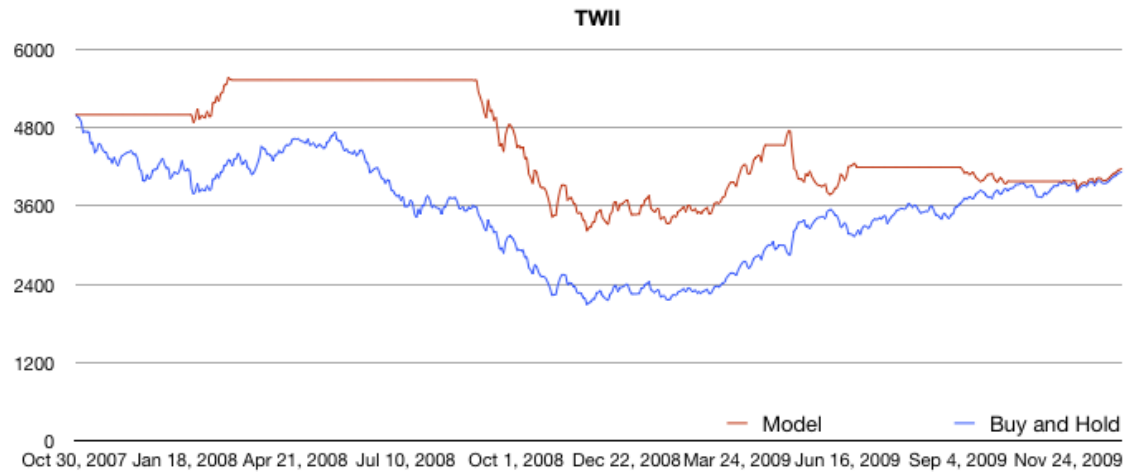

Figure 12: Simulation GP – TWII: 819 loss over 3 years.

4.5.2 Discussion

As can be seen, the simulation model has a hard time to beat the buy-and-hold benchmark. This is quite a substantial discovery. A lot of previous work relied solely on the static in-sample and out-of-sample returns as an indicator of performance, comparable with the results in Section 4.2 to Section 4.4. But as these results indicate, actually trading with a rule that fits a certain validation period beautifully, may prove to be not very profitable at all. This is especially true when the upcoming period has very low correlation with the previous validation period.

The simulation is mostly passive for the DJIA dataset, not doing anything until halfway in. This looks to be a good decision, as the market was mostly moving sideways. A correct short decision was made around July 2008, and things look good from that moment on, but unfortunately another short decision was made around April 2009. A net profit was realized though, and the benchmark has been beaten.

Even though the SP500 simulation did not manage to achieve higher profits than the buy-and-hold benchmark, the graph actually looks really good until the end. While not profiting fully from the uptrends, it was mostly spared from the downtrends. Unfortunately, a lot of money was lost at the end due to a couple of bad decisions at the end of the time period.

The TWII graph starts out nicely, mostly passive due to fitness and validation restrictions. At some point, a wrong buy decision followed by more bad decisions ruin the performance compared to buy-and-hold, resulting in a net loss, but still slightly beating the buy-and-hold benchmark.

Two out of three simulations do speak favorably for active management. Especially when taking into account that money could be made profitable when the simulation is passive (e.g. during the first 2 years of the DJIA simulation). This could be done by buying other bonds, savings accounts, or possibly other stocks.

4.6 Real-life Simulation: With ES

This final experiment is based on 4.5, but with the ES enabled. However, the ES is not used to optimize the top $n\%$ of the population after every round as in Section 4.4. This would take weeks to calculate on an average computer. Instead, the ES is used to optimize the original best ruleset and the potential new best ruleset after each day in the simulation.

However, instead of optimizing for the fitness period, the ES will be applied to the entire time period. Earlier experiments showed that optimizing for the training period leads to heavy overfitting. The same is true for optimizing only the validation period.

Additional measures to reduce overfitting have been applied: the minimum fitness requirement for trading is ignored and the GP split of 70/30 for training/validation was changed to 50/50.

Everything else is unchanged. See Figure 9 in Section 4.5 for a graphical representation of the simulation workflow.

4.6.1 Results

Dataset	Annualized Return	Sharpe Ratio	Abs. Return	Abs. Return BnH
DJIA	−1.6%	−0.39	−302	−761
SP500	−0.5%	−0.22	−327	2444
TWII	24.6%	0.67	2575	−865

See Figures 13, 14, 15 for graphs of the the model's cash value over time, compared to buy-and-hold, starting at 5000.

4.6.2 Discussion

With the ES enabled, performance has decreased for the DJIA and SP500 datasets, but has surprisingly increased a lot for the TWII data set. Of course, these results are just from a single run, but multiple runs support this finding.

The DJIA graph shows some good and bad decisions, and a couple of missed opportunities. Still, active trading was a better option than buy-and-hold in this case, but it does result in a net loss.

For the SP500 dataset, the simulation failed to profit from the rises from 1996 to 1999 and 2002 to 2007. Fortunately, it did not lose too much when the market fell after those rises. All in all, not much has happened during 13 years, with a total return of -327.

The TWII dataset graph shows very good conditioning. It nets a very respectable absolute return of 2575 (52%) over just 3 years.

Overall, again, two out of three simulations beat the buy-and-hold benchmark. Since multiple simplifications and assumptions are made during this research, these results do not formally disprove the Efficient Market Hypothesis. However, the results seem to hint at the possibility that it is indeed possible in some situations. More research (or an actual real-life test) would be needed to fully simulate a real life situation without simplifications.

5 Practical Implications and Assumptions

Simulations of complex markets always go hand in hand with certain assumptions and simplifications, and they are not missing from this paper. More research is needed before this model can be applied in a real-life setting. This section sums up all these assumptions.

Whenever a transaction decision is made on time t, with accompanying $price(t)$, it is assumed that the transaction can be executed without delay. In an actual market, the transaction would always be executed on time $t + d$ for $price(t + d)$. This difference in price may be relatively large, since in this case, the transaction is executed on the next day. Overnight market action should not be underestimated if there are late-night press releases or if there is other shocking economic news after the regional markets close.

The *spread* is the difference between the bid and the ask price. For some highly traded stocks, this is just a few cents, but for some less active stocks, this difference may be substantial. The spread is not taken into account in this research, and orders are assumed to be executed using the closing price. The closing price is usually an average of the bid and ask prices, or an average of the price of the last transactions.

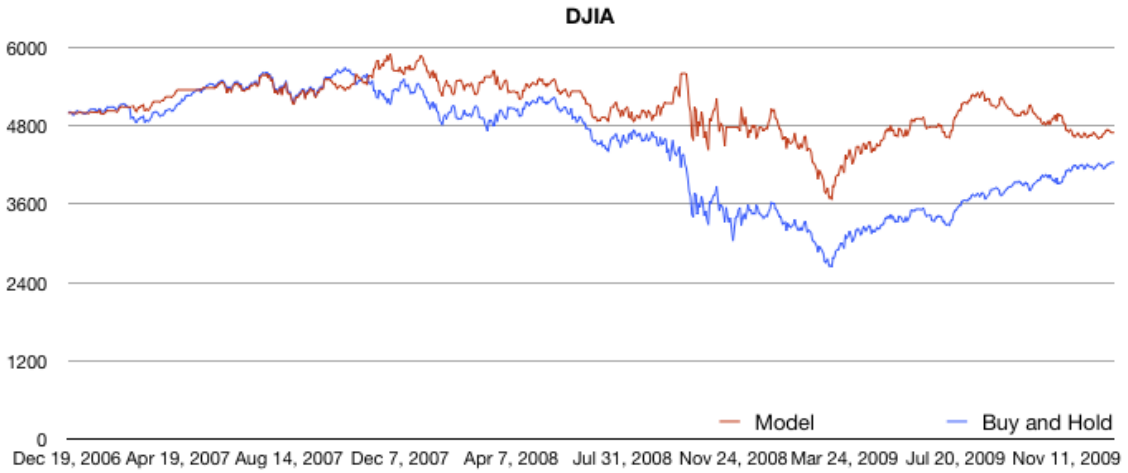

Figure 13: Simulation GP+ES – DJIA: 302 loss over 4 years.

Figure 14: Simulation GP+ES – SP500: 327 loss over 13 years.

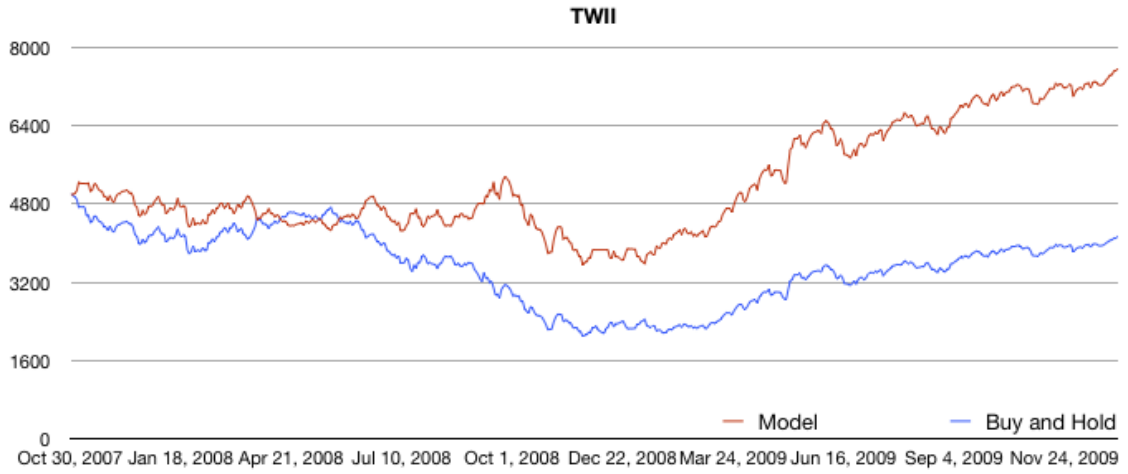

Figure 15: Simulation GP+ES – TWII: 2575 profit over 3 years.

It is assumed that the full amount of available money can be invested, instead of buying a fixed amount of stocks. This should make little difference when a high amount of money is available and the price of individual stocks is low enough, but is a simplification nonetheless.

Shorting stocks is usually done through the broker, and involves loaning the money to buy the stocks. There is a small amount of interest to be paid to the broker for this service.

Dividend payouts are ignored as well.

6 Conclusion

As shown by the experiments, trading rules built by GP are a powerful tool to encapsulate market strategies. High annualized returns can be reached if the stochastic GP process flips the coins favorably. Unfortunately, rules with high training and validation scores on a subset of a dataset do not generalize too well to unseen new data of that dataset. For example, on the SP500 data set, the GP with setsize 4 scored an annualized return of 9.8% out of sample on average when a static training and validation set were used (in Section 4.2.3). The experiment with the moving training and validation window tests the GP on consecutive subsets of the dataset (in Section 4.5) and the results are at a much lower 2.6%. The out of sample performance decreased by 73%. Results are lower for the other datasets as well.

One possible explanation is the overfitting that is caused by using absolute price and volume values in the decision trees. This is a problem, especially when the ES comes into play. Another risk of this structure is that subtrees in the form of `variable > constant` may be evolved, in which the variable may never reach the value specified by the constant again. For example, the price may never reach certain extremes again in the future. This problem may be solved by using a different grammar.

In the end, it remains to be seen whether this research can be used in a practical setting. The GP is a randomized optimization process which leads to a very rich set of solutions, as also indicated by the high standard deviation values in the experiments. But, sometimes, an above-average ruleset is evolved that could also be profitable on new data. However, as most investors know, excellent historic performance is hardly a guarantee for the future.

7 Future Research

The current objective function does not account for risks. Also, the function favors early gains over late gains, since those amplify the interest-over-interest effect. A better objective function could reward accumulation of wealth over time in a linear fashion. Incidentally, this also reduces the volatility in the portfolio, and therefore, the risk. This is something to work on for future projects.

Also, in order to fully harness the power of the ES optimization of constants, certain terminal variables could be replaced by others. Comparing price, volume and moving averages with constants is an absolute measure, that does not work well when the constants are optimized for a small subset of the data. Using the ES to squeeze out the maximum

performance for one time period is usually detrimental to the performance on another time period. A different grammar could be designed, which better allows to capture the underlying mechanics behind price movements. This grammar should allow for more fuzzy concepts, in lieu of the absolute `variable` $>$ `constant` constructs.

References

[1] F. Allen and R. Karjalainen. Using Genetic Algorithms to Find Technical Trading Rules. *Journal of Financial Economics*, 51(2):245–272, 1999. Elsevier.

[2] Thomas Bäck. *Evolutionary Algorithims In Theory And Practice*. Oxford University Press, 1996.

[3] W. Banzhaf, P. Nordin, R.E. Keller, and Francon F.D. *Genetic Programming*. Morgan Kaufmann, 1998. pages 119–120.

[4] L.A. Becker and M. Seshadri. GP-evolved Technical Trading Rules Can Outperform Buy and Hold. Technical report, Worcester Polytechnic Institute, 2003.

[5] Y.L. Becker, P. Fei, and A Lester. Stock Selection - an Innovative Application of Genetic Programming Methodology. Social Science Research Network, May 2006.

[6] S.H. Chen and C.H. Yeh. Toward a Computable Approach to the Efficient Market Hypothesis: An Application of Genetic Programming. *Journal of Economic Dynamics and Control*, 21(6):1043–1063, 1997. Elsevier.

[7] P. Cootner. *The Random Character of Stock Market Prices*. Cambridge: Massachusetts Institute of Technology Press, 1964.

[8] E.F. Fama. Efficient Capital Markets: A review of Theory and Empirical Work. *Journal of Finance*, 25(2):383–417, 1970. American Finance Association.

[9] E.F. Fama and M.E. Blume. Filter Rules and Stock-Market Trading. *Journal of Business*, 39(1):226–241, 1966. University of Chicago Press.

[10] C. Fyfe, J.P. Marney, and H.F.E. Tarbert. Technical Analysis Versus Market Efficiency - a Genetic Programming Approach. *Applied Financial Economics*, 9(2):183–191, 1999. Routledge.

[11] A.L. Garcia-Almanza and E.P.K. Tsang. Forecasting Stock Prices Using Genetic Programming and Chance Discovery. Computing in Economics and Finance 2006 489, Society for Computational Economics, July 2006.

[12] B. Graham and D. Dodd. *Security Analysis*. McGraw -Hill, 1934.

[13] N. Hansen, S.D. Müller, and P. Koumoutsakos. Reducing the Time Complexity of the Derandomized Evolution Strategy with Covariance Matrix Adaptation (CMA-ES). *Evolutionary Computation*, 11(1):1–18, 2003.

[14] Ghada Hassan and Christopher D. Clack. Robustness of Multiple Objective GP Stock-Picking in Unstable Financial Markets: Real-World Applications Track. In *GECCO '09: Proceedings of the 11th Annual conference on Genetic and evolutionary computation*, pages 1513–1520, New York, NY, USA, 2009. ACM.

[15] J.R. Koza. *Genetic Programming: On the Programming of Computers by Means of Natural Selection*. Cambridge: Massachusetts Institute of Technology Press, 1992.

[16] B.G. Malkiel. The Efficient Market Hypothesis and its Critics. *Journal of Economic Perspectives*, 17(1):59–82, Winter 2003. American Economic Association.

[17] D. Mallick, V.C.S. Lee, and Y.S. Ong. An Empirical Study of Genetic Programming Generated Trading Rules in Computerized Stock Trading Service System. In *5th International Conference Service Systems and Service Management-Exploring Service Dynamics with Science and Innovative Technology, ICSSSM*, 2008.

[18] John J. Murphy. *Technical Analysis of the Financial Markets*. New York Institute of Finance, 1999.

[19] C. Neely, P. Weller, and R. Dittmar. Is Technical Analysis in the Foreign Exchange Market Profitable? A Genetic Programming Approach. *The Journal of Financial and Quantitative Analysis*, 32(4):405–426, 1997. University of Washington School of Business Administration.

[20] J.Y. Potvin, P. Soriano, and M. Vallee. Generating Trading Rules on the Stock Markets with Genetic Programming. *Computers and Operations Research*, 31(7):1033–1048, 2004. Citeseer.

[21] P. Saks and D. Maringer. Evolutionary Money Management. In *Proceedings of the EvoWorkshops 2009 on Applications of Evolutionary Computing*, volume LNCS 5484, pages 162–171. Springer, 2009.

[22] P. Saks, D. Maringer, M. Giacobini, and A. Brabazon. Genetic Programming in Statistical Arbitrage. In *Proceedings of the EvoWorkshops 2008 on Applications of Evolutionary Computing*, volume LNCS 4974, pages 73–82. Springer, 2008.

[23] Massimo Santini and Andrea Tettamanzi. Genetic Programming for Financial Time Series Prediction. In *EuroGP '01: Proceedings of the 4th European Conference on Genetic Programming*, pages 361–370, London, UK, 2001. Springer-Verlag.

[24] W.F. Sharpe. Adjusting for Risk in Portfolio Performance Measurement. *The Journal of Portfolio Management*, 1(2):29–34, 1975. Institutional Investor Journals.

[25] B. Summers, E. Griffiths, and R. Hudson. Back to the Future: an Empirical Investigation into the Validity of Stock Index Models over Time. *Applied Financial Economics*, 14(3):209–214, 2004. Routledge.

[26] A. Timmermann and C.W.J. Granger. Efficient Market Hypothesis and Forecasting. *International Journal of Forecasting*, 20(1):15–27, 2004. Elsevier.

[27] CMA-ES: Covariance Matrix Adaption Evolution Strategy (Released under the GNU general public license). http://www.lri.fr/∼hansen/cmaesintro.html.

[28] L. Wagman. Stock Portfolio Evaluation: An Application of Genetic-Programming-Based Technical Analysis. Technical report, Stanford Universtity, Computer Science Department, 2003.

[29] W. Yan and C.D. Clack. Behavioural GP Diversity for Dynamic Environments: an Application in Hedge Fund Investment. In *Proceedings of the 8th Annual Conference on Genetic and Evolutionary Computation*, pages 1817–1824. ACM New York, NY, USA, 2006.

[30] W. Yan and C.D. Clack. Evolving Robust GP Solutions for Hedge Fund Stock Selection in Emerging Markets. In *Proceedings of the 9th Annual Conference on Genetic and Evolutionary Computation*, pages 2234–2241. ACM New York, NY, USA, 2007.

[31] Wei Zhang. Evolving Technical Trading Rules with a Hybrid Genetic Programming and Evolution Strategies Approach. Master's thesis, LIACS, Leiden Universtity, 2005.

Appendix

Best DJIA & SP500 Quad Rule Sets – evolved with GP

Best ruleset for the DJIA and SP500 datasets, see Section 4.2.3 for the experimental results. These rules are in the prefix notation.

DJIA

Buy long:

```
OR(OR(NOT(XOR(AND(OR(>(0.775681,RSI),<(0.0077941,RSI)),AND(>(MA_S,PRICE),
<(0.0225859,MA_L))),<(0.252738,MA_M))),AND(>(MA_S,PRICE),
<(0.0225859,MA_L))),NOT(XOR(XOR(<(0.625915,MA_S),>(0.482315,AROON)),
OR(>(0.775681,RSI),<(0.252738,MA_M)))))
```

Sell long:

```
NOT(AND(XOR(<(0.653126,AROON),NOT(<(0.653126,AROON))),
XOR(AND(<(0.653126,AROON),<(0.939834,AROON)),NOT(>(0.211857,VOL)))))
```

Buy short:

```
NOT(XOR(OR(OR(XOR(OR(<(0.375627,VOL),XOR(>(0.174557,RSI),
>(MA_L,0.147828))),XOR(>(0.174557,RSI),>(MA_L,0.147828))),>(0.174557,RSI)),
XOR(<(0.375627,VOL),>(0.238234,MA_L))),<(0.523383,MA_S)))
```

Sell short:

```
NOT(OR(NOT(OR(>(0.174971,RSI),>(RSI,0.733957))),
NOT(OR(NOT(OR(>(0.174971,RSI),>(RSI,0.733957))),>(0.563599,MA_M)))))
```

SP500

Buy long:

```
XOR(XOR(>(MA_S,MA_M),NOT(>(0.933329,MA_S))),<(PRICE,MA_M))
```

Sell long:

```
OR(OR(AND(OR(>(MA_S,MA_L),NOT(NOT(<(0.839316,PRICE)))),
NOT(>(0.97903,MA_S))),<(0.887489,MA_S)),AND(XOR(NOT(>(MA_S,MA_L)),
NOT(AND(NOT(<(0.737827,MA_S)),NOT(<(0.134982,RSI))))),AND(AND(<(MA_L,MA_S),
<(RSI,0.75639)),>(MA_M,0.650409))))
```

Buy short:

```
NOT(OR(OR(<(PRICE,MA_S),OR(<(PRICE,MA_S),OR(>(MA_M,MA_S),
AND(>(MA_S,0.31114),>(MA_S,MA_L))))),OR(>(MA_M,MA_S),AND(OR(<(PRICE,MA_S),
<(PRICE,0.963644)),AND(>(MA_S,0.31114),>(MA_S,MA_L))))))
```

Sell short:

```
AND(AND(NOT(<(0.383205,PRICE)),<(0.0176288,MA_S)),XOR(NOT(>(VOL,0.487411)),
NOT(AND(<(VOL,0.0698266),NOT(NOT(NOT(AND(>(AROON,0.766806),
AND(>(VOL,0.0698266),<(RSI,0.706313)))))))))
```