Genetic Algorithms on Technical Trading Rules

Silke Hofman (348261),

Abstract

In this paper, well know technical trading rules are used in different genetic algorithms to increase the performance on the S&P 500 exchange traded fund traded on the NASDAQ. Moving average rules, support and resistance rules, filter rules, momentum in price rules and channel breakout rules are combined to create one optimal rule to use on this data. The genetic algorithm creates an initial population of random rules and updates his population untill a population with this best rule is reached. The updating is of the same principle as Darwin's survival of the fittest. In this way, less performing will not survive through the next generation and better performing rules are. With Genetic Programming, an extension of the GA developed by Koza (1992), another type of optimal rules will be created. We will check whether this method creates better trading rules than our previous work and will compare several different ways to create these rules, when we end with a best rule for our data.

1 Introduction

In technical analysis, many methods and strategies are being used to attempt to forecast the direction of the market. These methods and strategies can be constructed by using past prices or other observable market statistics. Although technical analysis has been used and researched for years, there still is the discussion whether it is useful or not. In early studies, tests determining the performance of trading rules often indicated these rules consistently did not make any profits. Other studies, such as the studies of Lukac and Brorsen (1990) found that "several systems did generate returns significantly above the transaction costs." The problem is that when evaluating historically successful trading rules using historical data, researchers might be data-snooping. In this paper, genetic algorithms will be used for generating optimal technical trading rules on the S&P 500. Advantages of the genetic algorithm are that we can find ex ante optimal trading rules from certain primitive arithmetic operators. Because of this the biases we face when applying technical trading rules can be avoided.

This paper is an extension on Chung et al. (2013). In this paper, many different trading rules were performed on the same S&P 500 index as will be used in this paper. This paper will extend Chung et al. (2013), in a way that it will combine these rules to try to get more better results and to avoid data-snooping. The combining of this rules will be done by the genetic algorithm, an algorithm based on the Darwinian principle of

evolution. In this algorithm, trading rules will be computed when, after this, an initial population of (combined) trading rules will be created. For the creation of new generations we will use the fitness function, based on the Sharpe Ratio. Based on the fitness of this population, a new population will be created, by recombination, reproduction and mutation. This will be done untill the stopcriteria are reached, than the program will be terminated and the results will be evaluated.
Important results in this paper are the evaluation of the new trading rules, generated by the genetic algorithm. We will see that indeed, they will. The combined results do perform better than our benchmark models, especially in the test period, but there will also be a difference in the performance of the algorithms, mutually.

There have already been several different papers on genetic algorithm in combination with trading rules. For example, Roberts (2005) created optimal trading rules with genetic programming on 24 commodities. Also Neely et al. (1997) performed genetic programming, but they did that on technical trading rules on different currencies.

This research is divided into six sections. In the next section the data, used in this research will be discussed. Section 3 describes the genetic algorithm and the use of its methods. This section also contains the trading rules on which the genetic algorithm will be applied. In Section 4, results of the results will be represented, followed by the conclusion in Section 5. Section 6 will be used to describe some possible extensions of this research.

2 Data

The dataset used in this research is the same as in our previous research, Chung et al. (2013). The dataset contained a sample running from the 1^{st} of April 2009 up to the 30^{th} of June this same year. In this sample, daily NASDAQ data feed files were included. The data in this files contains order level data as well as trade messages, administrative messages and the five best current bid and ask quotations. Different orders are messages to add an order of type A and messages to modify existing orders (types C, D, E, U and X).

Table 1: Descriptive statistics of the midquote, subsequent high and low, and ret_t, used in the genetic algorithms.

	mean	σ	median	skewness	kurtosis
SPY					
midquote	89.3828	3.7968	90.1950	-0.4379	2.2061
ssh	89.4466	3.7910	90.2500	-0.4368	2.2053
ssl	89.3186	3.8048	90.1400	-0.4387	2.2070
ret	0.0000	0.0013	0.0000	0.0458	40.7342

Table 1 provides some descriptive statistics of our data used, which is based on the

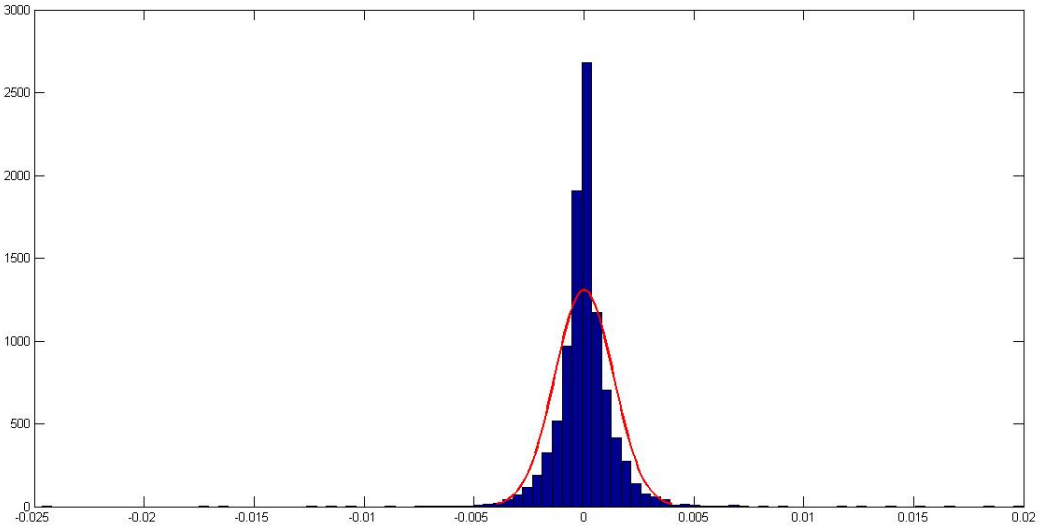

Figure 1: Histogram of ret_t

dataset used in Chung et al. (2013). The real dataset has already been discussed in our previous work, so for further information we refer to that paper. In this paper we will work with four sets of data, obtained from this dataset. First, we will use the midquote, represented by $m_t = \frac{p_{bid,t} - p_{ask,t}}{2}$. Also the subsequent high (ssh) and subsequent low (ssl) are used. The ssl and ssh are $max(m_{t_1} \dots m_{t-1})$ and $min(m_{t_1} \dots m_{t-1})$, respectively. At last, we will use $ret(x_t) = \frac{x_t - x_{t-1}}{x_{t-1}}$. All these four sets have already been used in Chung et al. (2013) and further discussed in that paper. In our algorithm, these four wil be represented by X1, X2, X3 and X4, in the same order as they are discussed before. In Table 1 we see that statistics are quite similar for the first three sets of data, but really different from the last set. This because all three sets are based on the same sample, in a same way and the last set in another way, by taking the first difference of the midquote and dividing it by the previous midquote. A Jarque-Bera test rejects normality in all four cases. The big value for the kurtosis combined with the relatively small skewness for the ret_t suggest really fat tails. Looking at Figure 1 this, indeed, is the case. While there are lots of observations around zero, there are few that are greater in absolute value. Figure 2 represents the midquote over time in days. Here, we see that in the first 50 days the midquote shows an upwarding trend, while in the last 13 days we see the midquote slightly declining.

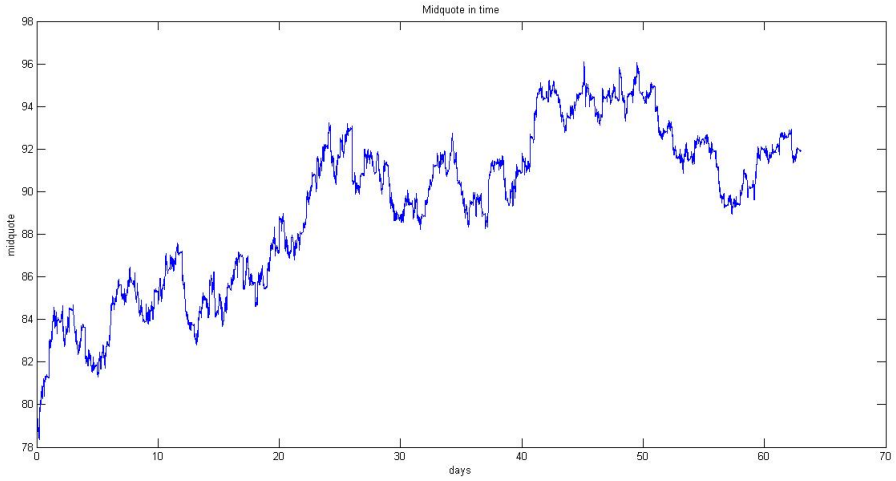

Figure 2: Midquote over time

3 Methods

The goal of this research is to generate optimal trading rules using a genetic algorithm, originally developed by Holland (1992). There are different kinds of genetic algorithms. In this paper we will use the conventional genetic algorithm, followed by the extended version 'genetic programming', based on Koza (1992). In this chapter, we will introduce the genetic algorithm in our trading-rule context, followed by some examples and a brief explanation of the algorithm. Then, we will get to genetic programming, based on Koza (1992) and explain why this is a good general apporach for this research.

3.1 Genetic algorithm

A genetic algorithm is an algorithm based on an early and well-known theory of Charles Darwin: evolutionary theory. The main idea of this theory is that in an initial population, there are 'losers' and 'winners'. The winners in a population are the fittest individuals, who are likely to survive and adapt to the current circumstances. Losers are the least fit individuals, who are likely to be extinct by these circumstances. Because our 'winners' are more likely to survive, they are also more likely to have children that will survive in these circumstances too. The whole idea of the genetic algorithm is exactly the same: We will start with an initial population, in our case this is a population consisting of trading rules. To create the next generation we are going to measure their fitness according to some fitness-function. With this fitness we will check whether individuals in the initial population will be reproduced, which means they will be copied into the next generation without any transformation, whether they will perform recombination, as in creating children, or whether they will be extinct.
Figure 11 in the Appendix shows a flowchart of the genetic algorithm. Figure 3 gives

an example of the representation of the trading rules in tree form. The first figure represents a simple moving average rule, while the second figure represents a filter rule. The moving average rule in the figure, creates a buy signal when the price exceeds the average of the 50 previous intervals. The filter rule creates the same signal when the price exceeds the minimum of the ten previous intervals multiplied with 1.01, so with a band of 0.01. In Figure 4, the recombination method is explained. Here, you see the two parents and one of the two children, called 'offspring'. The recombination method randomly searches through the rule part, when it finds a connector part, which is the part where two rules will be connected, it creates a head and a tail part from both of the parents. After this, the head part of the first parent, will be connected with the tail part of the second parents, and vice versa for the other child. This method will be used for a fraction of the total population. Another important method for creating a new generation is mutation. Mutation is an operation that aims at diversity in a population by applying random modifications to individual structures. Mutations will be performed only on the operators.

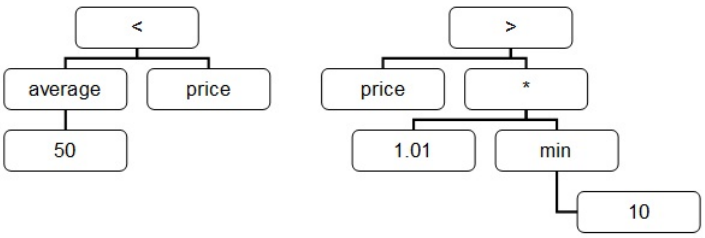

Figure 3: Trading Rules, the first rule is a simple 50-interval moving average rule and the second rule is a representation of the filter rule.

In the conventional genetic algorithm we are going to use, the individuals in a population will be represented by fixed-length strings with logicals. Let's say we have two trading rules, an individual in the population can be represented in the following way: 0 01 1 11. In this representation the first four logicals represent the indicator/connector/rule part, while the last two represent the structure part. The structure part tells you which rules are active within this individual. The first part creates the rule, where the second and third bits are the connector part, which connect the first indicator to the other with the following representation: 01 represents 'or', 00 represents 'and', and 10 represents 'xor'. So in this example, if in the first trading rule the signal is neutral or in the second rule the signal is 'buy', the combined trading signal is a 'one'. To get a more visual idea of how the trading rules are represented, we will explain it with a figure. In Figure 4, both parents consist of a combination of two rules. Let's say, these four rules are the only rules in the population and are indicated as rule A through B. Rule A of parent one is the first rule, rule B is the second. Now, in our genetic algorithm, parent 1 will be represented by the following combination of logicals: 1 00 1 ** 0 ** 0 1100, where the last four logicals represent the active rules in this individual: Only the first two rules are

included, represented with a '1' on the first and second place, and a '0' at the other two places. The ** are irrelevant, as the two other rules are not included in this individual and we will not create offspring out of these parts. Parent 2 is represented as follows: 0 ** 0 ** 1 01 1 0011, in the same way.

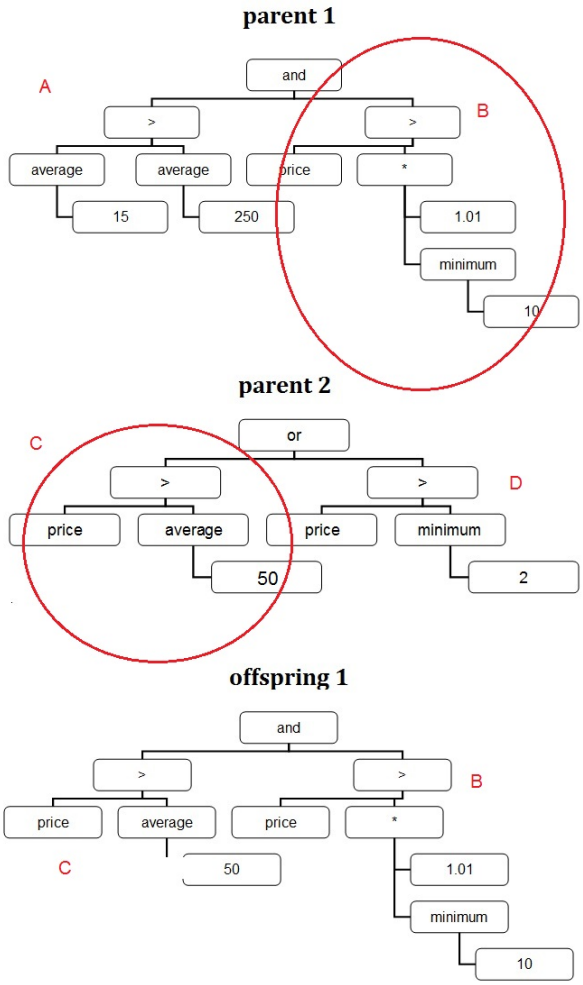

Figure 4: An example of a recombination, only one offspring is showed.

When we have finally created a whole generation of this logical strings, or individuals, we can calculate their fitness measure, which is based on the functions used in Chung et al. (2013):

$$r_{buy,t1,t2} = \frac{p_{bid,t2} - p_{ask,t1,imp}}{p_{ask,t1,imp}} \tag{1}$$

$$r_{sell,t1,t2} = -1 \times \frac{p_{ask,t2,imp} - p_{bid,t1}}{p_{bid,t1,imp}} \tag{2}$$

with $p_{bid,t,imp}$ ($p_{ask,t,imp}$) the *price-impact* bid (ask) price at time t. With this measure as a return we will calculate the Sharpe Ratio and our final goal is to maximize this Sharpe Ratio.

3.2 Genetic Programming

As was already mentioned, genetic programming is an extension of the genetic algorithm, mainly created by Koza (1992). The most important difference with the conventional genetic algorithm is the output of the algorithm. Where in the conventional algorithm the population was represented by a set of logicals, the output in genetic programming is a program itself. In this way, there are less limitations for the program to compute, what will probably lead to more strange rules, instead of a simple known Moving Average Rule.
There are lots of settings in genetic programming. We will run a maximum of 200 generations and there will be an initial population of 2000 individuals, just as in the genetic algorithm. We will start with a maximum depth of 10, which will avoid bloat. There are two different types to construct a tree, the first one is the Full Method. The Full Method is the standard procedure, where the tree receives new non terminal nodes until the maximum tree depth is reached. There is also the Growth Method, where terminal nodes and non-terminal nodes are chosen randomly, which leads to very unbalancing trees. We will start with the Full method.
For creating a tree we need some functions and terminals. A terminal does not have any subnodes and is the last node of a tree, mostly represented by a real constant. In this research terminal nodes are the midquote, subsequent high, subsequent low and ret, given by the function: $ret(x_t) = \frac{x_t - x_{t-1}}{x_{t-1}}$. Functions in this research are divided into certain groups, according to Potvin et al. (2004). There are arithmetic operators ($+, \div, -, \times$), boolean operators (and, or, xor[1], nand), relational operators ($<$, $>$), boolean functions (if-then-else) and real functions, represented by the average of one of the terminals over n days, the maximum of one of the terminals over n days and the minimum of one of the terminals over n days. Because we want to create a vector of zeros and ones, representing a buy or a sell signal, we cannot have every operator on each place as is the case in the normal genetic program. We have to add some restrictions for getting the correct output. The root can contain only boolean functions or operators and its descendants may only exist of boolean functions and operators or relational operators. As is the case for the genetic algorithm, we will construct new generations with crossover, reproduction and mutation. Each with fraction 0.6, 0.3667 and 0.0333 respectively as was found as the 'ideal fractions' by Papadamou and Stephanides (2007). We have also tried some other fractions, but their fractions performed better in all cases. Also the fitness function remains the same.We will measure our individuals with the Sharpe Ratio and the function in the previous subsection. Also selection of the individuals to perform crossover, mutation or reproduction on, happens in the same way.

[1]xor, or 'exclusive or', is true if exactly one, so not both, of towo conditions is true.

3.3 Rules and settings

For the genetic algorithm we are going to use five different sets of rules. These sets of rules consist of the same type of rules for each set, only with different settings, so that there are many rules to work with. Also the settings in for the genetic program are based on these rules. The five types chosen are the Moving Average (MA), the Momentum in Price (MP), the Filter (FI), the Support and Resistance (SR) and the Channel break-out (CBO) strategies.
The Moving Average rule is represented as follows:

$$MA_t = \begin{cases} 1 & \text{if } ma_t(s,m) > (1+b) \cdot ma_t(l,m) \\ -1 & \text{if } ma_t(s,m) < (1-b) \cdot ma_t(l,m) \\ 0 & \text{otherwise} \end{cases} \tag{3}$$

Where s represents the length of the shorter lookback period and l the length of the longer lookback period. Both the s and the l variables, as the b will be the same as in Scholtus and Van Dijk (2012). The Momentum in Price rule looks like the Moving Average rule, with one difference that not the moving average of the midquote will be taken, but the moving average of $ret(x_t) = \frac{x_t - x_{t-1}}{x_{t-1}}$. So $mar_t(l,x) = ma_t(l, ret(x))$. The MP rule will be represented by the following function:

$$MP_t = \begin{cases} 1 & \text{if } mar_t(s,m) > (1+b) \cdot mar_t(l,m) \\ -1 & \text{if } mar_t(s,m) < (1-b) \cdot mar_t(l,m) \\ 0 & \text{otherwise} \end{cases} \tag{4}$$

The next rule we will use is the Support and Resistance (SR) rule. For this rule we will need a maximum and a minimum midquote value over a lookback period of l intervals while we will account for intra-interval maxima and minima. The maximum (minimum) during an interval will be denoted as m_t^{max} (m_t^{min}), while the maximum (minimum) over a lookback period will be defined as $lbmax_t(l,m) = max(m_{t-1}^{max} m_{t-2}^{max} \dots m_{t-l}^{max})$.

The Support and Resistance rule generates a sell signal when the price penetrates the local minumum (support level) and a buy signal when the price penetrates the local maximum (resistance level). For these local maximum and minimum there are several different lookback periods, which can be found in the Appendix. There are also different bands implemented here. The SR rule will be given with the following formula:

$$SR_t = \begin{cases} 1 & \text{if } m_t > (1+b) \cdot lbmax_t(l,m) \\ -1 & \text{if } m_t < (1-b) \cdot lbmin_t(l,m) \\ 0 & \text{otherwise} \end{cases} \tag{5}$$

The fourth type of rule will be the Filter Rule (FI). This rule does not only use the midquote, but also the subsequent high and subsequent low, ssh and ssl respectively. A buy signal will be given when $ret_t > x$, while a signal of the opposite sign will be given when $ret_t < -x$. When none of these options is the case, we will look at FI_{t-1}. When

the previous signal is a buy signal and $\frac{m_t - ssh}{ssh} < -(x \cdot y)$ or the previous signal is a sell signal and $\frac{m_t - ssl}{ssl} > (x \cdot y)$, FI_t will give a neutral signal, otherwise FI_t will be equal to the previous signal given.

$$FI_t = \begin{cases} 1 & \text{if } \frac{m_t - m_{t-1}}{m_{t-1}} > x \\ -1 & \text{if } \frac{m_t - m_{t-1}}{m_{t-1}} < -x \\ 0 & \text{if } FI_{t-1} = 1 \text{ and } \frac{m_t - ssh}{ssh} < -(x \cdot y) \text{ or} \\ & \text{if } FI_{t-1} = -1 \text{ and } \frac{m_t - ssl}{ssl} > (x \cdot y) \\ FI_{t-1} & \text{otherwise} \end{cases} \tag{6}$$

The last rule to be constructed are the Channel Break-Out rules (CBO). This type of rule will use a channel to indicator to check whether there will be a signal at all. This channel, $I_{CH}(t)$, takes value 1 if there exist a channel and 0 otherwise. At time t a channel does exist if $lbmin_t(l, m) > y \cdot lbmax_t(l, m)$, where $y < 1$. A buy (sell) signal will be provided when the midquote at time t exceeds (falls below) the upper (lower) bound of the channel.

$$CB_t = \begin{cases} 1 & \text{if } I_{CH}(t) = 1 \text{ and } m_t > (1 + b) \cdot lbmax(l, m) \\ -1 & \text{if } I_{CH}(t) = 1 \text{ and } m_t > (1 + b) \cdot lbmin(l, m) \\ CB_t & \text{otherwise} \end{cases} \tag{7}$$

The settings for this rules, who are exactly the same as in Scholtus and Van Dijk (2012) can be found in the Appendix. For the genetic algorithm, each rule is constrained to 50 nodes as in Roberts (2005). In the data section, we stated there are 63 trading days we have available to perform our research on. We will use the first 25 days to perform the genetic algorithm on. The last 38 days will be used to evaluate the best trading rule, obtained by the algorithm. The percentages are based on the work of Potvin et al. (2004). The best trading rule will exist of multiple trading rules, represented by the set of logicals as was discussed in the previous part of the methods section. This best trading rule will be applied on the last 60 percent and returns and the Sharpe Ratio will be obtained here. In this way, we can perform an out-of-sample test, to check whether the best rule, obtained by an earlier data set, will actually work on the next set of data.

For our genetic programming algorithm, we will use different operator functions which are presented in the Appendix.

4 Results

In this section, results from the genetic algorithm and the genetic program on technical trading rules in the previous chapter will be discussed. This section is divided in four subsections, Section 4.1 discusses the Sharpe Ratio and some statistical results from the three obtained 'best rules' and compares them to our benchmark model. Section 4.2 discusses the more detailed results on the rules itself, whereas Section 4.3 compares the

results of the different genetic algorithms. In Section 4.4, we will look if the a rolling window will outperform our previpous results.
As was already mentioned in the previous chapter, 25 days of our total data were used to perform our genetic algorithm and our genetic program on. Both algorithms are thus performed on 25 trading days, which means the remainder 38 days were used to evaluate the trading rules. We will use two benchmark models. Our first benchmark model, 'WECO', will be represented by the rule with the best Sharpe Ratio in the 5-minute interval in our previous paper Chung et al. (2013). The model that performed best in this paper was one of the Moving Average rules. Our second benchmark model, 'benchmark', will be based on a buy-and-hold strategy: at the beginning of the day, when our trading hours start, we will have a buy signal, when at the end we will sell. For each day. We will evaluate our results with the Sharpe Ratio, based on the same function as in Chung et al. (2013) for consistent evaluation. In the Appendix an initial rule is represented. In the trees shown in this section $X1$ represents the midquote, $X2$ and $X3$ represent the subsequent high and subsequent low, respectively. $X4$ will be the $ret(m_t)$ already discussed in the previous section.

4.1 Performance

In this subsection, the basic performance of the rules is being discussed. The Sharpe Ratio, mean, standard deviation and total returns will be represented in Table 2.
In this table, r represents the returns based on Equation 1 and 2. Based on the Sharpe Ratio, the worst performing rule in the test period is the best WECO rule, with a Sharpe Ratio of 0.1772. The best performing rule in the test period is the best GP rule with the growth method for creating trees. Followed by the GP rule with the full method and the GA rule. This was expected, because the GP rule that uses the growth method has the least restrictions for creating such a rule. Here, the depth can vary over subtrees, which leads to less balanced trees, but also a greater variation in a population. The GP rule that uses the growth method has some more restrictions, as it has the restriction to create a tree where every subtree has the same depth. After these two, we have the GA rule, which is a combination out of existing rules and, thus, has even more restrictions, namely the restrictions of the rules itself. The best WECO rule only has one way to create the rule, because it is only one rule. Based on restrictions, results point out in the direction of our expectations.
When we look at the evaluation period, results do not seem to follow the same pattern as in the test period. Here, the best WECO rule is the best rule, based on the Sharpe Ratio, and the best rule in the test period seems to be the worst performing rule in the evaluation period. This can be the cause of the rules in the GP being only based on the data in the test period, while the WECO rule is based on the whole period, as it was the best performing rule over the whole period in Chung et al. (2013). The cause of the GA rule now performing better than both GP rules can be that the GA rule is based on already existing technical trading rules, which are used and examined in other papers for years. The GP only creates a best rule based on the test period, which may have caused the rule to perform not that good in the evaluation period. What we have

Table 2: In and out of sample returns of the GA technical trading rules.

	μ_r	σ_r	$\sum r$	SH
benchmark				
test period	0.0051	0.0116	0.1286	0.4525
evaluation period	0.0004	0.0098	0.0166	0.0453
Best WECO rule				
test period	0.0007	0.0033	0.0164	0.2041
evaluation period	0.0003	0.0017	0.0108	0.1644
Best GA rule				
test period	0.0049	0.0147	0.1213	0.3379
evaluation period	0.0013	0.0118	0.0508	0.1146
Best GP full rule				
test period	0.0052	0.0081	0.1310	0.6626
evaluation period	0.0007	0.0138	0.0269	0.0520
Best GP growth rule				
test period	0.0067	0.0092	0.1685	0.7459
evaluation period	-0.0014	0.0140	-0.0545	-0.1037

This table shows the average daily return r, the standard deviation of the returns, σ, the total return at the end of the period and the Sharpe Ratio. Here, the benchmark is based on the buy-and-hold strategy and the best WECO rule is the Moving Average rule, that performed best in Chung et al. (2013). The GA rule is the best rule based on the Genetic Algorithm, which is based on combinations of different existing technical trading rules. The GP full rule is the best GP rule, where a tree has to exist out of subtrees with each the same depth and the GP growth rule does not have this restriction.

also already seen it that the test period shows a much steeper slope on average than the evaluation period, which may have caused this huge decline in performance for the GP rules, compared to the performance in the test period. The standard deviations of the returns are very close in the cases of the GA and both GP rules, while the standard deviation of the WECO rule shows a smaller standard deviation. This means the deviations in returns in our created rules are bigger than our constant, standard rule. When we compare our rules with our other benchmark rule, we see that both GP rules outperform the benchmark in the test period. The GA rule does not outperform our benchmark rule in the test period, but exceeds the performance of the benchmark in the evaluation period. The GP rule that uses the full method outperforms our benchmark in both periods, but the GP rule that uses the growth method does not outperform the benchmark in the evaluation period.
In the next subsection we will have closer look at the differences in performance and trading of our different rules.

4.2 Performance in detail

In this subsection, the best trading rules of the GA and GP's and the benchmark rule are evaluated more in detail. We will take a deaper look into the trading rules itself and will focus on a possible difference between the rules. Table 3 presents the more detailed results. The first column represents the fraction of time in the period the returns are greater than 0 and the last two columns give the number of buy and sell signals, each for the testing period and the evaluation period.

Table 3: In and out of sample returns of the GA technical trading rules.

	% r>0	# Long	# Short
benchmark			
test period	0.7143	25	0
evaluation period	0.5263	38	0
Best WECO rule			
test period	0.0005	1	0
evaluation period	0.0005	1	0
Best GA rule			
test period	0.6800	223	212
evaluation period	0.4737	381	309
Best GP full rule			
test period	0.7200	82	79
evaluation period	0.5526	112	83
Best GP growth rule			
test period	0.7600	69	56
evaluation period	0.5263	103	71

This table shows the average daily return r, the standard deviation of the returns, σ, the percentage of times the returns are greater than zero (r>0), and the percentage of long and short position are taken.

As we saw in the previous section, the total returns of the WECO rule performs worst for the test period. In Table 3 we see that there is only one trade in this period, a buy. Therefore, we are not surprised the fraction of times the rule generates returns above zero is very small (it is either one positive return or zero). For the rest of the rules, the fraction of times the returns are positive are also divided in the same way as the total returns in the previous section. So, the GP growth rule has the highest fraction of returns above zero, followed by the GP full rule and the benchmark. Finally, the GA rule and WECO rule follow.
Looking at the amount of buys and sells, we see that both the WECO rule only has one, while for the benchmark rule every day one buy signal is given. For both GP rules, the number of long and short positions are close together, but the GA rule has over twice as much as the GP rule. This could be the cause of the GA rule combining different rules, where the best rule is a combination of a couple of 'good buys and sells'. With

75 intervals per day, the amount of total buy and sell signals are not much more than 10%.

When we take a look at the evaluation period, based on the results in the previous chapter and the patterns in the test period, we would expect the fraction of returns greater than zero for the best GA rule to be the greatest (for the WECO rule, there is only one trade, so one positive return). This, in fact, is not the case. These fractions of all the rules, except the WECO rule are really close together. The reason for the GP growth generating negative results, therefore can be the cause of three events. The negative returns of the GP growth rule are more negative than the other rules, the positive returns are not as high as the other rules, or a combination of the two. In Figure 5, a graphical representation of the cumulative positive returns are presented. Figure 6 represents the cumulative absolute negative returns. In Figure 5 we see that both GP

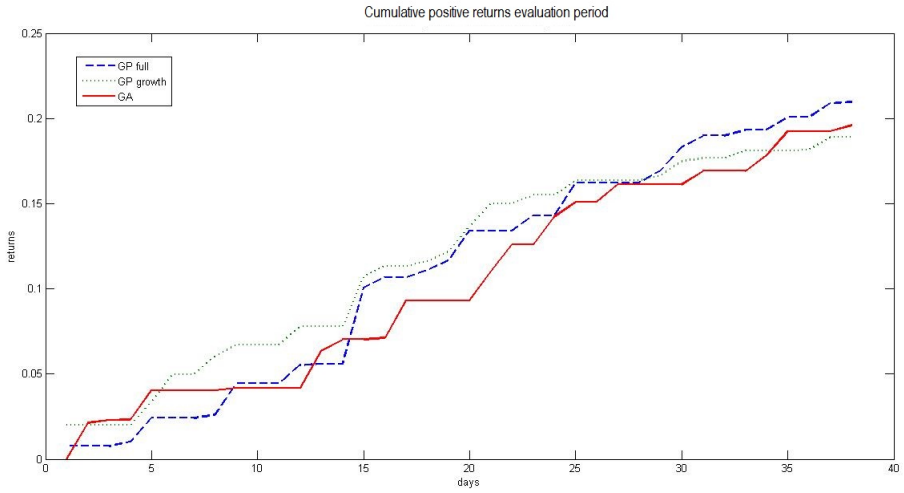

Figure 5: Graphical representation of the cumulative positive returns, respectively

rules show the same patterns in returns for the first 25 days in the evaluation period. In this period, the growth rule clearly performs better than the full rule. As we have already seen in the Section 2, after this day, the midquote is declining. In Figure 7 and 8 the rules are represented in tree-form. Here, we see that the growth rule generates a buy signal when the maximum of the ssh of the last 55 intervals of 5 minutes exceeds the average of the ssl the last eight intervals. Because prices are declining, a signal based on the last 55 days does not give an accurate image of the current prices. The rule now, gives a lot of buy signals, but when the position at the end of the day is closed, prices are still declining which leads to losses. This will go on until the end of the evaluation period, which results in a decline in the returns of the growth rule.
When we look at the full rule, this rule generates a buy signal when the minimum of the last 15 interval's maximum of the last 40 interval's of the midquote exceeds the midquote at that point in time. Clearly, this is a much more accurate image of the prices as it is based on only 15 interval's, instead of the 55 in the growth rule.

In Figure 6 again, we see similar patterns for the negative returns. Because the growth

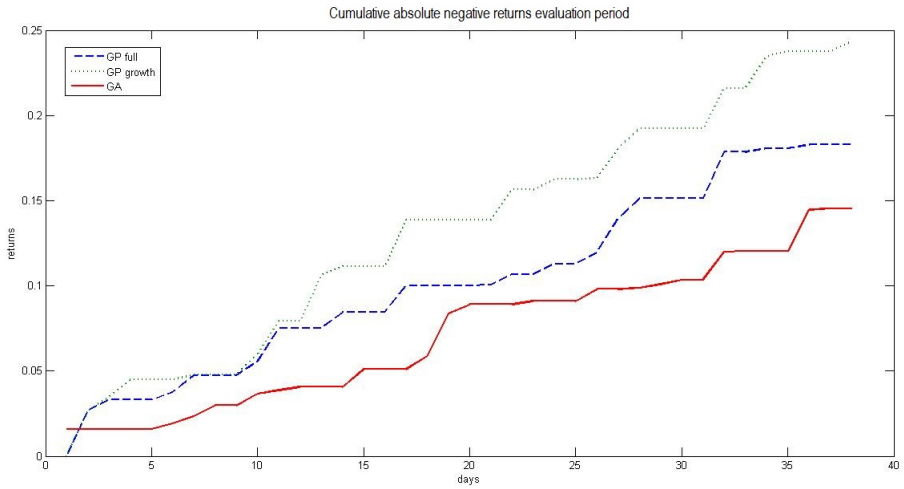

Figure 6: Graphical representation of the cumulative positive returns, respectively

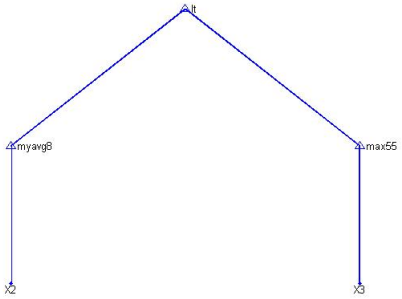

Figure 7: The best growth rule represented in tree-form

rule is less accurate than the full rule, more negative returns are generated, which also results in the total low returns in the evaluation period.

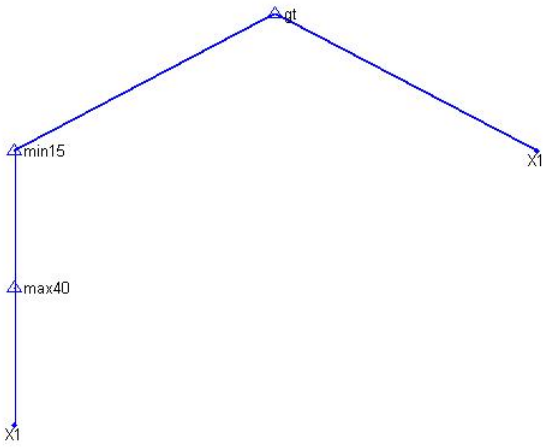

Figure 8: The best full rule represented in tree-form

4.3 Genetic Algorithm vs. Genetic Programming

In this subsection we will have a last look at the differences of the performance of the Genetic Algorithm with the performance of the Genetic Programming methods. We will look at the rules in a tree-form and will repeat our main observations. At last, we will have a look at the return patterns through time.

The best GA rule uses two MP rules, two MA rules, a FI and a SR. As was already mentioned, the growth rule is based on the last 55 intervals, which leads to a less accurate image of the prices at that moment. This could be the reason for the difference in performance at the end of the evaluation period. In the test period, the Genetic Program creates rules that perform best on that dataset. Because the market fluctuates a lot, the rules that first worked very well, now can become less performing rules. The Genetic Algorithm combines existing, and well researched technical trading rules into one rule, which can make the rules in general better substantiated.

The figures below represent the cumulative returns in the evaluation period. Here, both GP's show similar patterns, although the rules are clearly different. The best GA rule does not show such a pattern. This is because this rule is based on already existing rules and is created in another way than the GP rules, which have less restrictions in creating a tree.

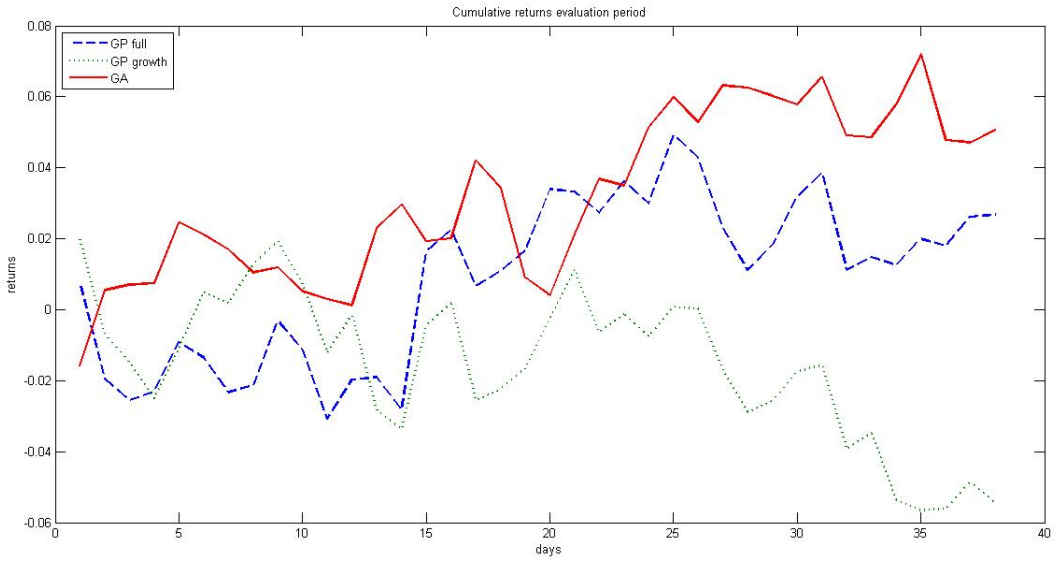

Figure 9: Cumulative returns evaluation period

4.4 Rolling window approach

In this section we are going to observe whether a rolling window will improve our previous results for the full method. The reason why we are only performing a rolling window on the full method, is because this method has least computational speed.
We use the rolling window only to create a best rule for the evaluation period, so, we begin at the 26^{th} day. For day 26, with the genetic program, we create a best rule out of the previous 25 days, so day 1, 2, ... , 25. For day 27 we do the same, so here, we create a best rule out of day 2 through 26. We do this for all 38 days, and based on these best rules for each day, we create signals for each day. With these signals we can calculate the fitness in the same way as in the previous sections. We use 25 days so that we can compare the results, obtained with the rolling window, with our previous results.

In Table 4, results for the GP full method with the rolling window are presented.

Table 4: In and out of sample returns of the GA technical trading rules.

	μ_r	σ_r	$\sum r$	SH	% r>0	# Long	# Short
GP full, rolling window evaluation period	0.0008	0.0125	0.0318	0.0680	0.6053	110	48

This table shows performance for the rolling window approach for 25 days for the GP using the full method.

Comparing these results with our previous results of the full method, we see that the rolling window outperforms our results without the rolling window. Although, results

do not outperform the GA and the WECO rule, we see that the fraction of times the daily return is positive is greater for the full method with the rolling window. Here, because of the rolling window, for day 63 we do not include the first 37 days, which leads to more accurate trading signals. Hence, in these last days, we do not include the, on average, increasing midquote in the first 25 days, which could have been the cause of the greater amount of positive returns. Also the standard deviation of the returns is less than for the full method without the rolling window, we now have less negative returns, which lead to less deviations in returns and, thus, a smaller standard deviation. The amount of buys is approximately equal to the amount without the rolling window, while the amount of short selling is less than before.
With the rolling window, we obtained better results than without the rolling window for the same method. As we have seen, our midquote in the first 25 days is, on average, increasing, while after these days its increase declines. By using the rolling window approach, a best rule for the last days is not obtained including these observations, which leads to better trading signals and, thus, results. Although, use of this method does not generate better trading rules compared to our other, better rules in the previous subsection, it does outperform the full rule without the rolling window. Maybe if we applied the rolling window to the other methods, we may have outperformed the WECO rule with our genetic algorithms.

5 Possible extensions

In this article, we used different kinds of genetic algorithms to obtain better technical trading rules than in our previour work on technical trading. With the rolling window approach for the full method, we found better results than for the full method without the rolling window. Due to time issues and a long computational speed (about one week for the rolling window on the full method), we were not able to extend our research by applying the rolling window on the other methods. A possible extension on this paper, thus, would be to apply the rolling window on all methods and then compare these results with our benchmark models and our former results.
Another way to extend this paper, is to add some more operators and functions, for example the first difference, or other functions, for more better results.

6 Conclusion

This research was an extension on the paper Chung et al. (2013). This paper extended the other paper in the way that it used the dataset of the paper and the technical trading rules described in it. The idea behind the GA and GP is the theory of Darwin, where the fittest individuals survive, reproduce and create offspring. The GP updated their trees, the outputs, in two ways. The first way was the growth method and the second way the full method. The Sharpe Ratio was used as the rules' fitness, where a higher Sharpe Ratio represented a better individual.With the Genetic Algorithm these rules

were combined to a population of combined rules, where the algorithm was searching for an optimal rule. In another way, the Genetic Programming method, an extended version of the Genetic Algorithm by Koza (1992), tried to create an optimal trading rule for this dataset. The difference with the GA was that the GP output was a function, in the form of a tree, where the GA gave a fixed output of zeros and ones, representing the combination of different, existing trading rules. The input of the GP were simply operators and terminals, where the algorithm created a random initial population and tried to create a better population. We measured their performance and compared them with the best rule in ourpervioud paper and added another benchmark model, based on the buy-and-hold method.
We saw that in the test period, the GP's performed the best in all cases, the GA and benchmark followed, and the worst performance was given by the rule from WECO. In the evaluation period, things change. The WECO rule now was the best performing rule, followed by the GA rule and the GP full rule. The GP growth rule performed worst, with a total negative return and Sharpe Ratio. The reason for the GP growth rule performing this bad, was that this rule is based on the previous 55 5-minute intervals, while there is a decline in the prices after 25 days in the evaluation period. The growth rule was not able to adapt this decline and produced 'bad buy signals', which led to a decline in positive returns and an incline in negative returns for this period. The cause of the WECO rule and GA rule performing this good may lie in their roots: both rules are existing rules, that have been used for any years in technical trading. The rules are already based on such a dataset as ours, while the GP rules found a best rule out of functions and operators for only the test period. Because of the decline of our dataset only in the evaluation period, the GP rules could not adapt that well, as they did in the test period. Our benchmark rule performed worse, because this rule is not based on any properties of the dataset itself. It is just buying at the beginning of the day and selling at the end of that same day. Again, because of the decline in price, buying is a less good idea, which leads to less positive returns.
As an extension, we also used a rolling window of 25 days on the full method. Here, results seemed to be a little better than the full method without the rolling window and also there were more positive returns than in all the other rules in the evaluation period.

We have learned a lot on the Genetic Algorithm and Genetic Programming, we saw that for the test period, they could lead to better trading rules than our already existing rules. Unfortunately, in the evaluation period they were not able to outperform our first benchmark rule, the WECO rule. The GA rule did outperform the other benchmark model in the evaluation period, because this rule was based on real technical trading rules, while the second benchmark model was not. The advantage of the algorithms were the fact that there is data-snooping, which we did find in our previous paper. Unfortunately, due to time issues, we were not able to extend our paper in the other ways, but when further research will be done, also these extension will be examined.

Acknowledgments

The authors of this paper would like to thank their instructor Martin Scholtus for providing very helpful and quick feedback throughout the drafting of this paper.

References

Chung, H. L., Appels, L., Huang, I. and Hofman, S. (2013), 'Technical trading'.

Holland, J. H. (1992), *Adaptation in natural and artificial systems: an introductory analysis with applications to biology, control and artificial intelligence*, MIT press.

Koza, J. R. (1992), 'Genetic programming: on the programming of computers by means of natural selection'.

Lukac, L. P. and Brorsen, B. W. (1990), 'A comprehensive test of futures market disequilibrium', *Financial Review* **25**(4), 593–622.

Neely, C., Weller, P. and Dittmar, R. (1997), *Is technical analysis in the foreign exchange market profitable? A genetic programming approach*, Cambridge Univ Press.

Papadamou, S. and Stephanides, G. (2007), 'Improving technical trading systems by using a new matlab-based genetic algorithm procedure', *Mathematical and computer modelling* **46**(1), 189–197.

Potvin, J.-Y., Soriano, P. and Vallée, M. (2004), 'Generating trading rules on the stock markets with genetic programming', *Computers & Operations Research* **31**(7), 1033–1047.

Roberts, M. C. (2005), 'Technical analysis and genetic programming: Constructing and testing a commodity portfolio', *Journal of Futures Markets* **25**(7), 643–660.
URL: *http://dx.doi.org/10.1002/fut.20161*

Scholtus, M. and Van Dijk, D. (2012), 'High-frequency technical trading: The importance of speed'.

Appendix A Visuals

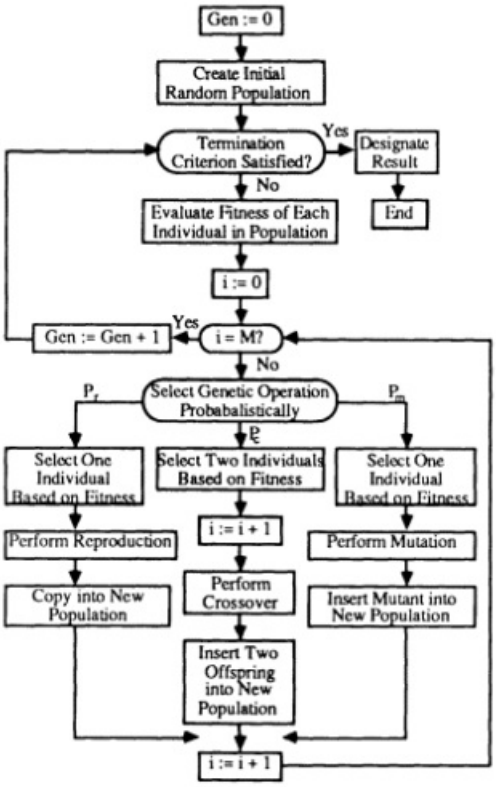

Figure 10: Basic idea genetic algorithm

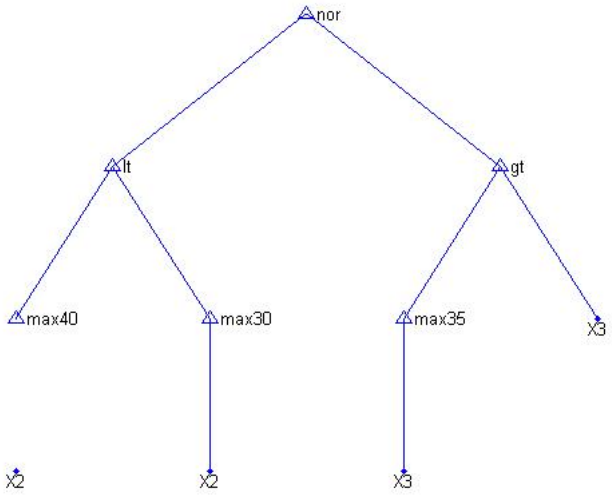

Figure 11: Random tree in the initial population

Appendix B Choice of parameters for constructing rules

Settings for the MA and MP rules, with l the long period, s the short period and b the used bands

l: 4,5,6,7,8,9,10,**11**,12
s: **2**,3,4,5,6,7,8
b: 0,0.00025,**0.0005**,0.00075,0.001,0.0025

For the best WECO rule, the bold numbers are the setting for the Moving Average equation.

Settings for Support and Resistance

l : 3,5,10,15,20,25,30,35,40,45,50,55,60
b: 0,0.00025,. . . ,0.001,0.0015,. . . ,0.0035,0.00375

Filter settings

x: 0.00025,0.0005,. . .,0.007
y: 1,0.75,0.5,0.25

Settings for the CBO rules

l : 5,10,15,20,25,30,35,40,45,60
y: 0.01,0.02,. . . ,0.1
b: 0,0.0001,0.00025,0.0005,0.001,0.00125,0.0015

Appendix C Operators in the genetic programming algorithm

Here, a short description of some operators and functions is given.

gt:	greater than
lt:	less than
myif:	if-then-else function. When if holds, we choose the 'then' input, otherwise we use 'else'.
myavg2 t/m myavg12:	moving averages from 2 intervals until 12 intervals, respectively.
mymax'no.':	the maximum of the last 'no.' of intervals.
mymin'no.':	the minimum of thelast 'no.' of intervals.
mydivide:	the quotient of two inputs.